NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

VRML TERRAIN MODELING FOR THE MONTEREY BAY NATIONAL MARINE SANCTUARY (MBNMS)

by

R. Greg Leaver

September 1998

Thesis Advisor: Don Brutzman
Associate-Advisor: Rex Buddenberg

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1998	Master's Thesis

4. TITLE AND SUBTITLE : VRML Terrain Modeling for the Monterey Bay National Marine Sanctuary (MBNMS)

5. FUNDING NUMBERS

6. AUTHOR
R. Greg Leaver

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(maximum 200 words)***

This thesis develops an online model of the topographic terrain of Monterey Bay National Marine Sanctuary (MBNMS) seafloor. Written in the Virtual Reality Modeling Language (VRML), the model is an interactive 3D application composed of hundreds of topographic tiles linked together to form a mosaic of the bay. Low-resolution tiles are traded for higher resolution tiles as the viewer gets closer to the terrain.

Important contributions include a naming convention for autogeneration of interlinked files, test usage of proposed metadata conventions linking VRML and the eXtensible Markup Language (XML), demonstrated use of the GeoVRML Working Groups proposed QuadLOD node, and a preliminary 3D navigation icon for terrain interrogation and wayfinding. Terrain data was produced from registered, smoothed and subsampled bathymetric sonarscan results. Because the model is geo-referenced with the Universal Transverse Mercator (UTM) coordinate system, a user can easily add scientific content or data to a selected location of the MBNMS in a manner analogous to adding 2D content to an HTML page. Thus, the user can place 3D content anywhere in the MBNMS in geographic context merely by specifying the geographic coordinates and depth of the content in standard VRML syntax.

Future work includes improvement of metadata interoperability, navigation icon user testing, and autogeneration of image-based texture tiles for scientific visualization.

14. SUBJECT TERMS
World Wide Web, Virtual Reality Modeling Language (VRML), Large-Scale Virtual Environments (LSVEs), Monterey Bay, 3D Graphics Modeling

15. NUMBER OF PAGES
126

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

VRML TERRAIN MODELING FOR THE MONTEREY BAY NATIONAL MARINE SANCTUARY (MBNMS)

R. Greg Leaver
Lieutenant, United States Navy
B.S., Oklahoma State University, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1998

Author: _____
R. Greg Leaver

Approved by: _____
Don Brutzman, Thesis Advisor

_____
Rex Buddenberg, Associate Advisor

_____
Reuben T. Harris, Chair
Department of Systems Management

ABSTRACT

This thesis develops an online model of the topographic terrain of Monterey Bay National Marine Sanctuary (MBNMS) seafloor. Written in the Virtual Reality Modeling Language (VRML), the model is an interactive 3D application composed of hundreds of topographic tiles linked together to form a mosaic of the bay. Low-resolution tiles are traded for higher resolution tiles as the viewer gets closer to the terrain.

Important contributions include a naming convention for autogeneration of interlinked files, test usage of proposed metadata conventions linking VRML and the eXtensible Markup Language (XML), demonstrated use of the GeoVRML Working Groups proposed QuadLOD node, and a preliminary 3D navigation icon for terrain interrogation and wayfinding. Terrain data was produced from registered, smoothed and subsampled bathymetric sonarscan results. Because the model is geo-referenced with the Universal Transverse Mercator (UTM) coordinate system, a user can easily add scientific content or data to a selected location of the MBNMS in a manner analogous to adding 2D content to an HTML page. Thus, the user can place 3D content anywhere in the MBNMS in geographic context merely by specifying the geographic coordinates and depth of the content in standard VRML syntax.

Future work includes improvement of metadata interoperability, navigation icon user testing, and autogeneration of image-based texture tiles for scientific visualization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

To my wife Sandra, my daughter Lauren, and my son Chase, thank you all for supporting me in this endeavor. I love and treasure each of you more than mere words can say. To Don Brutzman, I offer my thanks for your contagious inspiration, enthusiasm, and guidance. To Ray McClain, I am indebted to you for your assistance in this project. I couldn't have done it without your help.

I. INTRODUCTION

A. BACKGROUND

This thesis investigates how the Virtual Reality Modeling Language (VRML) can be used to model the seafloor topography of the Monterey Bay National Marine Sanctuary (MBNMS). By creating a topographic model of the MBNMS using VRML, a three-dimensional representation of the sanctuary can be accessed over the World-Wide-Web (Web) by anyone using a VRML-enabled web browser or standalone VRML viewer. A VRML-enabled browser means a browser configured with a VRML plug in such as Cosmo Player for PCs (Silicon Graphics, 98). Rapid recent progress in this field means that many new opportunities are available.

B. MOTIVATION

Numerous scientists and researchers are collecting data and building environmental models about Monterey Bay. Regional research partnerships using a Large Scale Virtual Environment (LSVE) for Monterey Bay will make it easy for scientific content about Monterey Bay to be placed and accessed online. Building a Monterey Bay terrain model is a dramatic way to encourage scientists to their work in three-dimensional (3D) space and on the Web. New insights and new research collaborations are likely. A new paradigm for publication of scientific data and analytic results is possible.

C. OBJECTIVES

The goal is to make the addition of a user-selected portion of MBNMS terrain in a 3D VRML scene as easy as adding a background image to a 2D HTML page. Thus, it is

hoped that this effort will make it easy for scientific content about Monterey Bay to be placed and accessed online, in a 3D geographic context.

Although many scientists are conducting research in Monterey Bay, bathymetric terrain scenery is not easily available. VRML makes 3D graphics accessible to any desktop. Constructing a LSVE for Monterey Bay may dramatically enhance ongoing regional research collaborations. An additional objective is for the model to support variable resolutions of gridded data. "Variable resolutions" essentially means that as a viewer gets closer to terrain, the resolution of the terrain increases to provide superior granularity.

D. THESIS ORGANIZATION

The remaining chapters of this thesis are organized as follows. Chapter II provides the background for the effort, introduces a few cartographic concepts, and touches on some related work being done. Chapter III presents the problem statement and covers design considerations for a feasible solution. Chapter IV provides a look at bathymetric data sources and describes the gridding process used to create simple gridded text data files. Chapter V shows how these elevation grid terrain text files can be processed by a Java program to produced VRML world files. Chapter VI discusses the specific VRML constructs produced by the Java programs that implement the MBNMS terrain model, including 3D navigation/information icons. Chapter VII considers experimental results, examines user access, and shows how users can integrate their 3D content and MBNMS terrain. Chapter VIII presents thesis conclusions and provides recommendations for future work.

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

This chapter examines pertinent background work that motivated the construction of a terrain model for the MBNMS and introduces the basic concepts of VRML. It also discusses other work being done to produce 3D topographic models, and provides a quick look at a tool evaluated by the author that creates 3D topographic scenery in VRML from elevation data sets.

B. BACKGROUND

1. Monterey Bay National Marine Sanctuary

Beginning 11 km north of San Francisco's Golden Gate Bridge, the MBNMS[1] extends 260 km south along the California coast to Cambria Rock in San Luis Obispo County. The Monterey Bay National Marine Sanctuary contains the nations greatest diversity of marine life and habitat. East to west, the sanctuary stretches 152 km and holds one of the world's largest ocean canyons: the 10,663 ft. deep Monterey Canyon. Thus, this area provides unparalleled opportunities for marine scientists based at nearby research institutions such as Moss Landing Marine Laboratories (MLML), Monterey Bay Aquarium Research Institute (MBARI), and NPS. Figure 2.1 illustrates the MBNMS region. The sanctuary was established to enhance resource protection and preserve the natural beauty within its boundaries.

[1] More information on the MBNMS is available at *http://bonita.mbnms.nos.noaa.gov/*

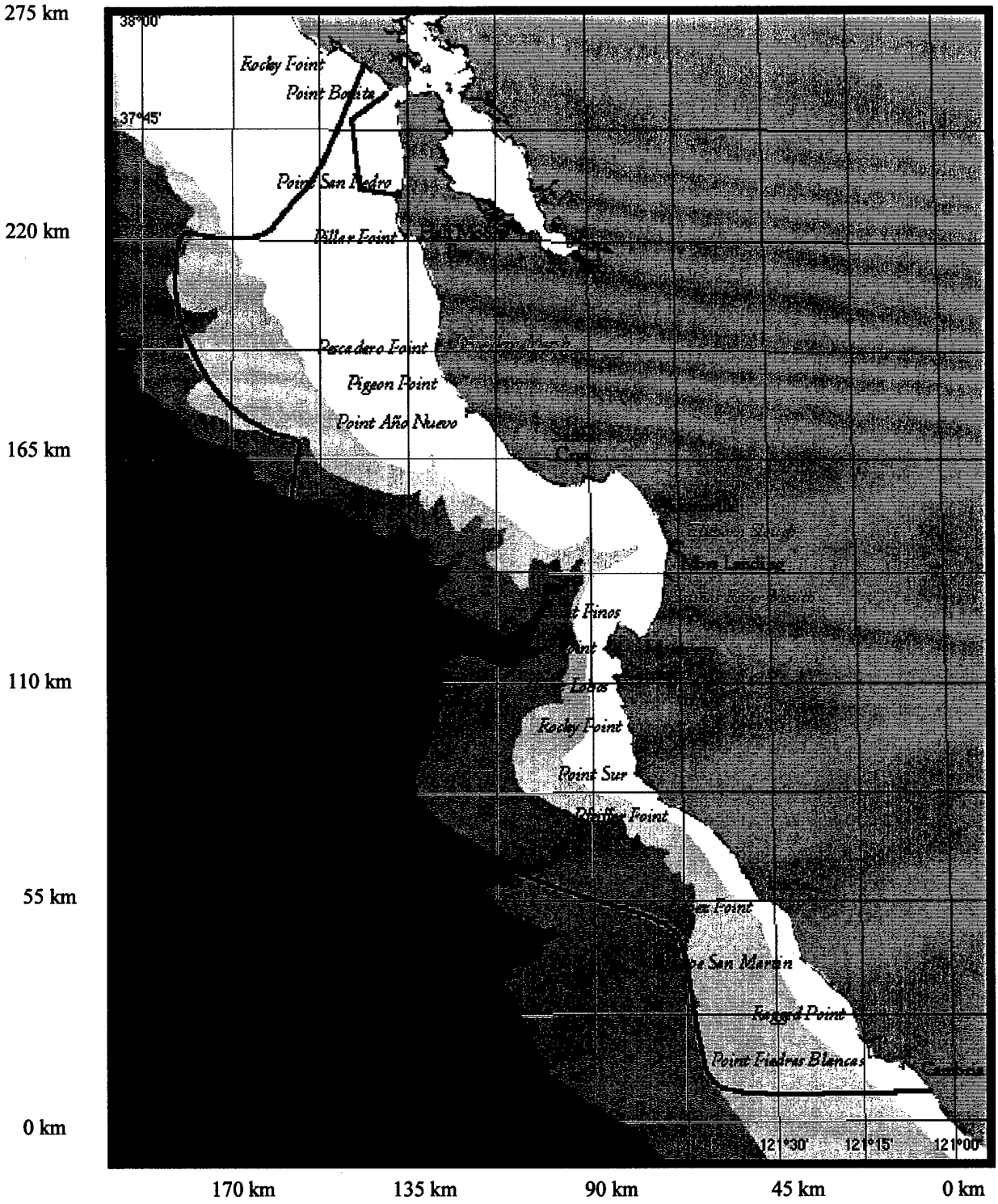


Figure 2.1. Monterey Bay National Marine Sanctuary (MBNMS Web Site, 98)

2. Monterey Bay Modeling Group

Interest in developing computerized processes and models to assist with studying and managing the sanctuary led to the formation of the Monterey Bay Modeling Group. In 1993, the MBNMS Research Advisory Committee, under the sponsorship of the National Oceanic and Atmospheric Administration (NOAA), prepared a research plan which outlined the research priorities and management goals for the sanctuary. This plan outlined the objectives of the Monterey Bay Modeling Group, an ad hoc group of individuals interested in computer modeling and affiliated with various MBNMS research organizations, including NPS. Listed among the objectives was the goal for the development of a computerized model of the sanctuary (NOAA, 93). The model, it was hoped, would ultimately function as an oceanographic scientific database archival and retrieval system, which could be overlain on a 3D physiographic representation of the MBNMS. The model would be networked for use by scientists, engineers, planners, managers, and the general public. Unfortunately this group was only active for two years. Recent discussions indicate that technology has advanced sufficiently to enable further scientific collaborations.

3. Coordinate Systems Used

In terms of latitude and longitude, the MBNMS occupies a square region between 35°30' North and 38° North latitude, and 123°15' East and 121° East longitude. Since latitude and longitude are commonly used measures, they are included in the model's metadata and file-naming convention. Universal Transverse Mercator[2] (UTM) coordinates, which specify a location as a distance north (Northing) and east (Easting)

[2] A good explanation of the UTM coordinate system is available at: *http://geography.tqnm/msub14.htm*

from a zone's meridian measured in meters, are also frequently used in cartography. In terms of the UTM system, the MBNMS lies between Northing coordinates of 3,940,000m to 4,200,000m, and Easting coordinates of 460,000m to 612,000m. Because UTM coordinates are in meters rather than degrees, UTM measurements can easily be converted to VRML coordinates that are default units in meters. This capability allows gridded elevation data and scientific content to be positioned relative to their real world location in the MBNMS model.

4. What is VRML?

VRML - the Virtual Reality Modeling Language - is a 3D graphics scene description language that enables a scene builder to create dynamic worlds and sensor rich virtual environments on the Internet. VRML enables users to animate objects in worlds, making them move; it also enables users to play sounds within worlds, interact with worlds and, control and enhance worlds with scripts, or small programs (Ames, et. al., 97). VRML provides a standardized, portable, and platform-independent way to render dynamic, interactive, 3D scenes across the Internet (Brutzman, 97).

A VRML file generally ends with extension ".wrl". This file is a textual description of a 3D world. A VRML file contains nodes that describe shapes and their properties in the virtual world. These nodes make up the building blocks - VRML constructs - which create the 3D scenery in a virtual world. For cartographic models such as this project, one of the principal VRML constructs is the *ElevationGrid* node, which can be used to create a 3D representation of the terrain. The terrain itself is described by a data set containing bathymetric depth values. Each sampled depth value is

associated with a pair of gridded 2D coordinates. An excellent overview of how VRML can be applied to cartography can be found in Fairborn and Parsley (97).

C. RELATED WORK

1. GeoVRML Working Group

To provide a forum for discussions of the representation and exchange of properly geo-referenced data in VRML advance, the GeoVRML Working Group was established. One of the forum's goals is to establish VRML as a standard for the representation and exchange of 3D geographic and cartographic data. The GeoVRML mailing list is maintained as part of the GeoVRML Working Group of the VRML Consortium by SRI International. Figure 2.2 shows the primary issues of interest and concern of the working group.

- **Coordinate systems** - measurement systems including the Geodetic and Geocentric systems used to specify locations on the surface of the Earth.
- **Time referencing** - important for content that is timestamped with respect to a an absolute reference.
- **Terrain representation** - imagery usually represented in an array of numbers that represent topography in digital form.
- **Levels of detail** - the hierarchy of resoltions necessary to achieve acceptable rendering and performance for a LSVE
- **Resolution and accuracy** - factors limited by georeferencing VRML worlds to a coordinate system and by data storage issues.
- **Data interchange** - standardized data format and type to enable data exchangge and interoperability.

Figure 2.2. Focus of GeoVRML Working Group (Iverson, 98)

2. Seamless Solution's Terrain Navigator

The Terrain Navigator is implemented entirely in Virtual Reality Modeling Language (VRML) for use on a low-cost Personal Computer (PC) to enable a content developer to integrate highly realistic terrain content with Web pages. This real-time

interactive visualization software is especially useful for collaborative review of database development throughout the design cycle or for entertainment purposes. (Seamless Solutions, 98).

3. SIGGRAPH CARTO Project

The "Carto Project" began in 1996 as a cross-organizational collaboration between the activities of the Association for Computing Machinery's Special Interest Group on Graphics (ACM SIGGRAPH) and the International Cartographic Association's (ICA) Commission on Visualization. The Carto Project explores how viewpoints and techniques from the computer graphics community can be effectively applied to cartographic and spatial data sets. This includes exploring how viewpoints and methods from cartography can enhance developments in computer graphics; especially those associated with the representation of geographic phenomena. These efforts will continue into 1999, in conjunction with the time frame of the ICA's Commission on Visualization (Rhyne, 98).

4. VRML Terrain Generators

Several commercial products exist that can automatically generate VRML terrain. To do so, generally these products import a dataset in a prescribed format such as Digital Elevation Model[3] (DEM) and produce export a VRML file via a filter. Rapid Imaging Software offers a product called LandForm Gold (RIS, 98) that works like this. A copy of this software was evaluated by this author, courtesy of Mike Abernathy at RIS. LandForm Gold is a powerful 3D real-time terrain viewer for the Windows NT/95 platform. This product allows a user to view geographical data in a three-dimensional

[3] DEM files contain data of the elevation of the terrain over a specified area, usually at a fixed grid

representation and move through the data in a natural and intuitive manner. The program accepts numerous file types and allows the user to superimpose an image of the area over the terrain. This effect of the image overlay combined with 3D data creates a strikingly realistic representation of the terrain, as landmarks and topographical features are dramatically revealed in 3D. As mentioned earlier, LandForm Gold also enables the user to create VRML models based upon the dataset read by the viewer. Other tools, such as Cybertrek (98) and Coryphaeus (98) are also available to create VRML terrain models, but were not evaluated by this author.

5. Synthetic Environment Data Representation & Interchange Specification (SEDRIS)

The SEDRIS Geographic Reference Model (GRM) has been proposed by the GeoVRML Working Group (discussed later) as a standard for VRML coordinate systems. SEDRIS is a reference model and software package that currently supports 12 different commonly used world coordinate system convention, as well as tools to automatically convert reference marks between them. Coordinate system standards supported include Geodetic (GDC or latitude/longitude), Geocentric (earth centered Cartesian), Universal Transverse Mercator (UTM), and Lambert Conformal Conic (LCC). The proposal (GeoVRML, 98) was drafted by SRI International and is summarized here. It proposes two levels to employ VRML constructs that implement the SEDRIS standards. Level 1 consists of a means of entering geographical coordinates into VRML files so that the Cartesian VRML coordinates are generated with respect to a geographically referenced local coordinate system. Its use depends only on the

---

interval, such as 1 arc-degree or 7.5 arc-minutes. DEM files are avilable (for a fee) from the U.S. Geological Survey at: *http://edcwww.cr.usgs.gov/webglis*

availability of a library for converting from geographical coordinates in the GRM into a local Cartesian frame. Level 2 consists of an attempt to establish a means for automatically managing the relationships between the local Cartesian frames defined in Level 1. It is intended as the enabling technology for seamlessly integrating accurately georeferenced worlds from a wide variety of sources. Since the constructs contained in the proposal are experimental at this time, they were not employed in the MBNMS Model. Nevertheless they remain an important area for future work.

6. Other Existing VRML Terrain Models

SRI International has developed a VRML terrain model of the Fort Irwin, California area. This terrain model has been distributed on CD-ROM and is also viewable on the Web (SRI, 98). It uses multiple levels of detail to change the terrain's resolution based upon the viewer's distance to the scenery. In 1997 RIS produced a model of the San Francisco Bay area (Abernathy, 98). This model was produced to convey topographical information to participants in the San Francisco Relay. By integrating Global Positioning Satellite elevation data with satellite and aerial imagery, the model displayed the terrain the event's course and scenery from a runner's point of view.

A simple textured model of Monterey regional terrain is also available at *http://ece.uwaterloo.ca/vrml98*. It provides background for the VRML 98 Symposium 3D website

D. SUMMARY

This chapter explores the related work that preceded or motivated the creation of a model for the MBNMS. It places the sanctuary in a geographical context and presents

the goals of the Monterey Bay Modeling Group. A brief overview of VRML provides some basic concepts of this scene-description language. Work related to VRML terrain development is considered and VRML terrain authoring tools are introduced - one of which is evaluated by the author. Additionally, some existing VRML terrain models are identified.

III. PROBLEM STATEMENT

A. INTRODUCTION

Although excellent commercial software exists for professional development of topographic models (both VRML and non-VRML) in 3D, these tools can be expensive and may require data sets to be in a proprietary format. Furthermore, the tools are not only necessary to generate the topographic models; they are also often required to be present on a user's console in order to view the models. By representing the model in VRML, an open solution to the problem of generating topographic data sets and subsequently viewing them is obtained. Anyone with a web browser and WWW access can potentially interact with the model. This chapter covers the problem of developing such an application. It then discusses the advantages of VRML as the basis for implementing a solution. Much of the chapter is devoted to an examination of the design issues considered in the development of a model solution.

B. RESEARCH FOCUS

In the last few years, advances in 3D modeling languages have made it feasible to develop the foundation for a 3D model of the MBNMS. VRML in particular enables the development of a model that can be viewed over the WWW on any platform with Internet access and a VRML enabled browser. In order to place arbitrary research information about the MBNMS in a geographic context, a user needs to be able to relate MBNMS data with the location the data describes or pertains to in the sanctuary. By constructing the model in VRML, a background of the appropriate geographic location can be added in a manner analogous to adding a background texture to a 2D HTML page. VRML can be geo-referenced with a cartographic coordinated system to support the

addition of the terrain to a scene. The goal of this thesis is the development of a VRML-based model of the MBNMS that is accessible over the Web which can enable a user to select a portion of Monterey Bay National Marine Sanctuary (MBNMS) terrain and easily add scientific content or data to the selected scenery. Thus, the end product will allow the user to place 3D content in a geographic MBNMS context. Of primary concern when attempting to construct such a model is the methodology by which low-resolution 3D scenery is exchanged for higher resolution scenery, all the while maintaining the overarching context of the MBNMS environment with the content added. Because bathymetric data of the MBNMS seafloor is available and that data can be gridded to various resolutions, and because the model needs to be based upon open-standard VRML that is directly authorable, commercial development tools are not required for this effort.

C. DESIGN CONSIDERATIONS

In addition to being accessible via the Web, a VRML model of the MBNMS needs to be navigable and geographically accurate. The units of measurement should be in meters and coordinates specified where possible to foster cartographic relation and interpretation. The process of switching between resolutions is an important consideration that leads to the introduction of two important terms: *tiles* and *transitions*. The term *tiles* refers to a VRML file covering a certain sized horizontal area that can be exchanged for another tile or group of tiles at a different resolution. The term *transition* describes the method of the actual change, i.e., the mechanics of the switch. Thus a tile undergoes a transition from one resolution to another. The method used for transiting between resolutions has dramatic performance implications.

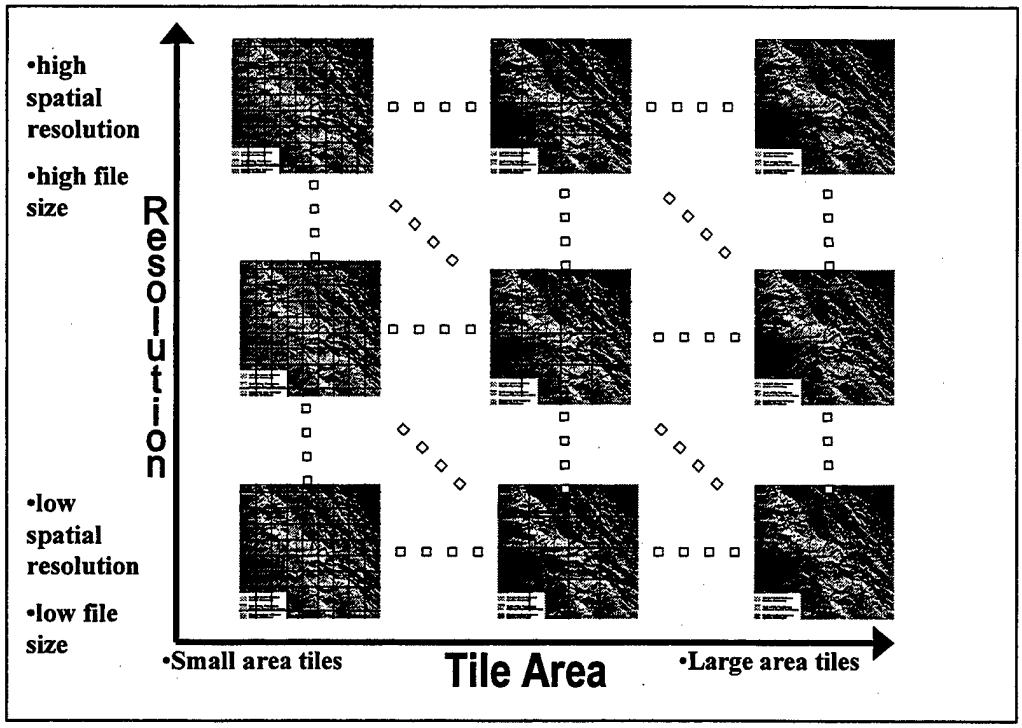


Figure 3.1. Tile Transitions

1. Transitions Considered

The effectiveness of transitions to switch between higher and lower resolution tiles is not just a function of the mechanism used. It is also dependent upon file download time (which is a function of file size and bandwidth) and platform rendering speed. Figure 3.1 illustrates the transition schema. To minimize bandwidth requirements, multiple resolution tiles arranged in a tree hierarchy can be used. Thus, only low-resolution tiles need to be loaded initially. As the viewer gets closer to a region of interest, the lower-resolution tiles are switched out with higher-resolution tiles, which are downloaded as needed. Minimal level of detail (resolution) also aids in platform-rendering speed. Transitions can be considered to be of three types: "SwapTile," "QuadTile" and "QuadSwapTile." The three types are illustrated in Figure 3.2.

a. "SwapTile" Transition

This transition swaps a single low-level tile of given area for a single tile of twice the resolution with the same area. Thus, there is a one-for-one tile exchange when this transition occurs.

b. "QuadTile" Transition

This transition trades a low-level tile of given area for four tiles (a quad) that are each the same resolution as the parent tile. The total area of the four tiles covers the same area as the single, previously viewed tile.

c. "QuadSwapTile" Transition

This transition trades a low-level tile of given area for four tiles that are twice the resolution of the parent tile. In other words, a quad of high resolution tiles switches out one low-resolution tile. This transition is equivalent to doing a combination of the other two transitions, i.e., "QuadSwapTile" = ("SwapTile" + "QuadTile"). Nevertheless, a SwapTile + QuadTile combination is less efficient than a QuadSwapTile transition since additional intermediate files must be loaded.

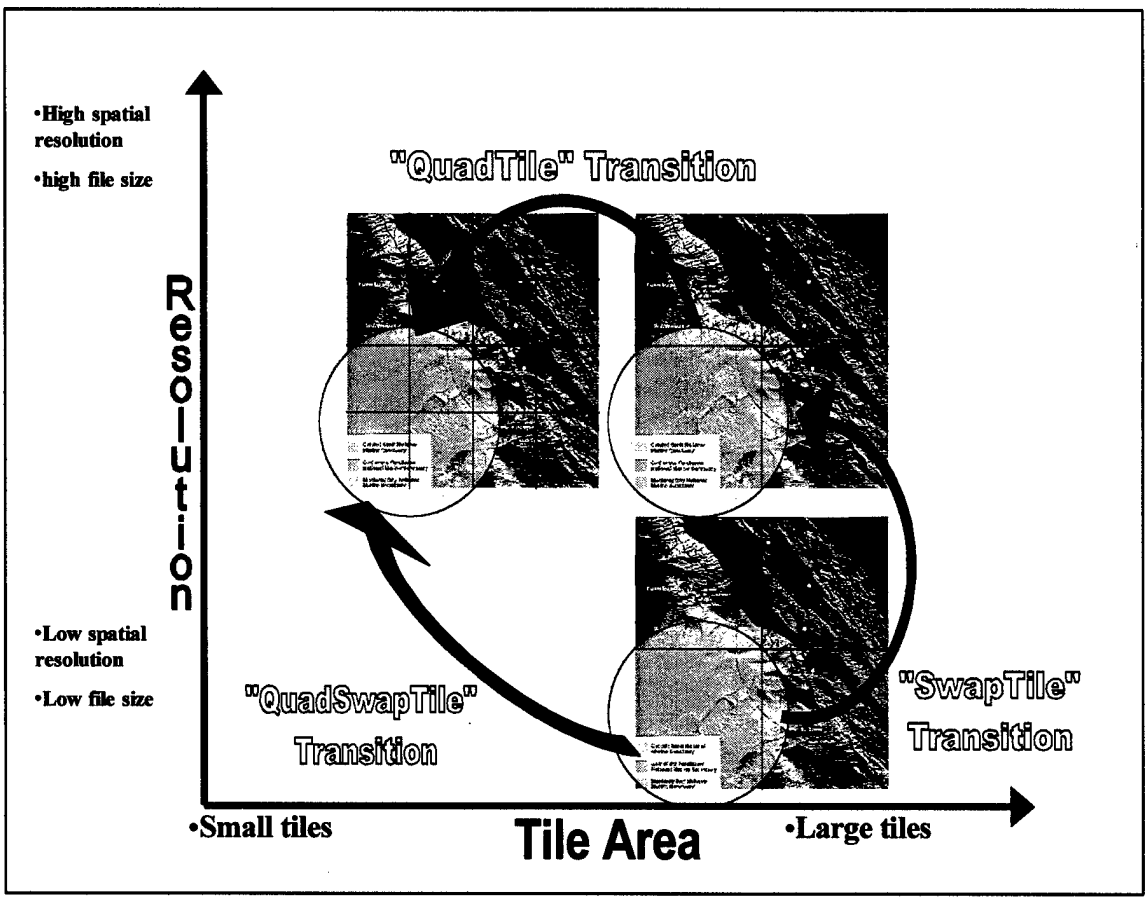


Figure 3.2. Tile Transition Types

If the user's goal is to gain successively higher resolution, note that the direction of the transitions is important. As a region of interest is approached, transitions will tend to be toward higher grid/file resolution in combination with smaller tile subdivisions, i.e., upward and leftward on the above two charts. Alternatively, as a region of interest is exited, transitions will be downward and rightward as a region of interest is exited.

2. Transition Chosen

The "QuadSwapTile" transition was chosen for the implementation of this project because its result is a combination of the other two transitions and it offers a scalable means of implementing a hierarchy of switching trees. Its concept is also supported by an existing VRML construct (discussed in Chapter VI). Thus it can be readily

implemented once the terrain tiles are developed. Figure 3.3 displays the design of the project using the "QuadSwitch" transition. Nevertheless, the other two transitions, the QuadTile and the SwitchTile, remain useful (at least theoretically) to support higher-resolution rendering on higher-performance machines, with the dataset quickly retrievable via dedicated Internet connections or local disk availability. Future work to integrate all three transitions is discussed in Chapter VIII.

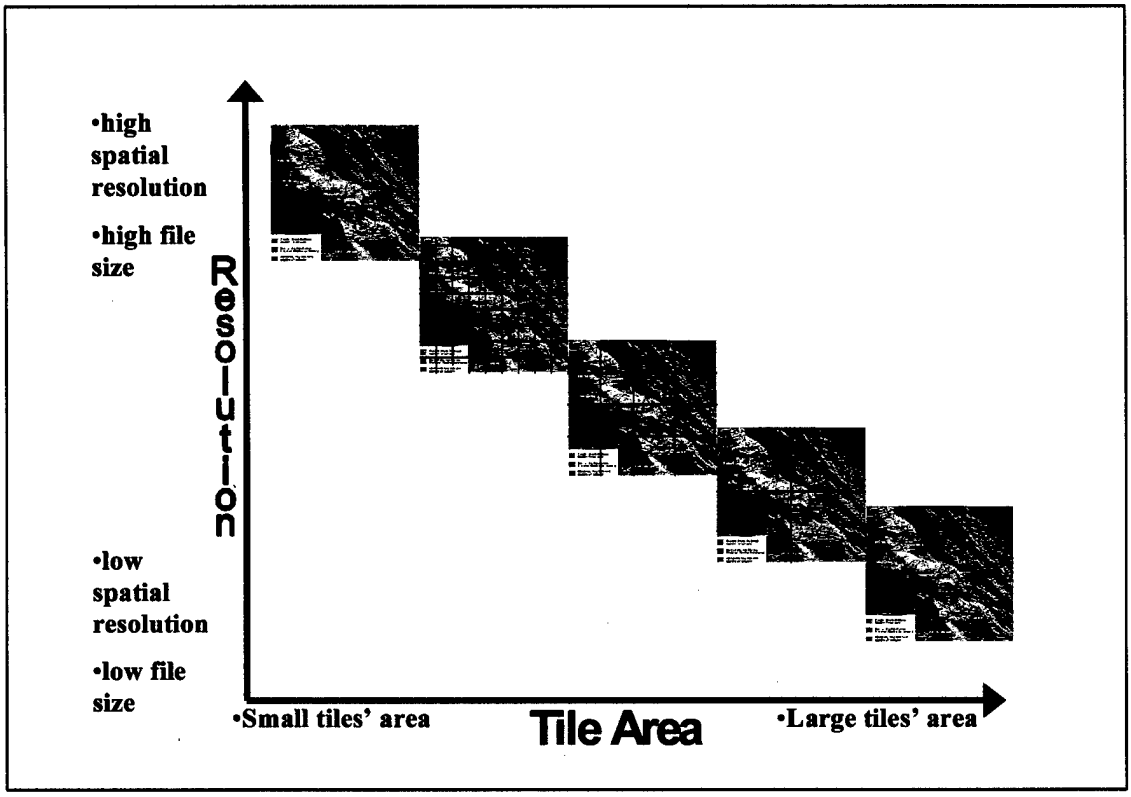


Figure 3.3. Implementation of QuadSwapTile Transition

D. SUMMARY

This chapter describes the nature of the problem that this thesis addresses. It discusses some of the advantages to using VRML as the scene description language to implement a solution and some of the reasons why existing tools are not required. It examines terrain tile-swapping characteristics of design in detail.

IV. BATHYMETRIC TERRAIN DATA

A. INTRODUCTION

This chapter discusses the data used in the MBNMS Terrain Model. It covers the source and processing of the data. It explains how raw, ungridded, topographic data is gridded to desired resolutions and how monolithic grids are partitioned into smaller sections. It describes how file names are built for gridded data files and explains the metadata included with those files.

B. DATA PROCESSING

1. Data Source and Gridding Process

Ungridded bathymetry data, which is simply raw elevation data of the seafloor topography, has data points at intervals determined by the sampling rate of various oceanographic surveys. The data points equal the depth of the ocean at that sampled spot. The database containing the ungridded bathymetric is maintained by the U.S. Geological Survey (USGS). The USGS obtained the data by way of the Seabeam-sonar oceanographic survey done by the National Oceanic and Atmospheric Administration. Moss Landing Marine Laboratories (MLML) obtained a copy of the data (MLML, 98) which is now public domain (USGS, 98).

In order to be useable for a MBNMS Terrain Model, the raw data needs to be gridded at an evenly spaced interval (post-spacing) such that each data point is a fixed distance from each neighbor. Posts refer to the notion that elevation heights at regular intervals are similar to posts of different heights spaced throughout a flat field. Thus, the data points form a grid at a given resolution. The closer the points are to one another, the higher the resolution of the grid. The raw, ungridded dataset for MBNMS comprises one

large 40 MB file. This dataset can be gridded to various resolutions using the Kriging Algorithm.[4] Using this algorithm as implemented in a GIS program named "Surfer for Windows," (Golden Software, 98). Ray McClain of MLML gridded the entire dataset at post-spacing-interval resolutions of 2000, 1000, and 500 meters, respectively. Appendix A contains scripts for batch files that accomplish this. This effort effectively created three new datasets, each describing the entire seafloor topography of the MBNMS at the respective resolutions (McClain, 98).

2. Partitioning the Datasets

Each of the five gridded datasets can be divided into partitions. Each partition contains bathymetric data describing a piece of the MBNMS at the gridded resolution. The original coordinate system used in Seabeam Survey data is in decimal degrees (latitude/longitude) format. However, the coordinate system needed in the final partitioned files is UTM. To perform this conversion, MLML developed a script (Appendix B) that creates a batch file to input into a geographic information system (McClain, 98). This GIS, called "TNTmips" (MicroImages, 98) used the batch file to perform the coordinate system conversion and partitioned the datasets as necessary. Table 4.1 shows the outcome of the partitioning process.

[4] Kriging is a method of interpolation which predicts unknown values from data observed at known locations. This method minimizes the error of predicted values which are estimated by spatial distribution of the predicted values. MLML used a Linear Variogram model with no anisotropy, no drift, and a zero nugget effect.

Resolution (Post-spacing) of dataset	Number of partitions in dataset	Number of posts in partition	Partition file-size in kilobytes (KB)	Aggregate MBNMS Data Set Size in Kilobytes (KB)
2000m	1	10,087	90KB	90KB
1000m	4	10,087	90KB	360KB
500m	16	10,087	90KB	1,440KB
Totals	21	All have 10,887 posts	All file sizes are 90 KB	1,890 KB

Table 4.1. File Partition Characteristics by Resolution for Entire MBMNMS Footprint

3. How Resolutions Were Determined

The three resolutions (2000m, 1000m, and 500m) were determined by considering the interplay between file size, maximum dataset size and post-spacing resolution intervals. Note in that in Table 4.1, above, the file size for any one partition is constant since the number of posts in all partitions is the same. This occurs because as postspacing is halved, the surface area covered by a partition drops to one-fourth the area of the previous, higher resolution partition, keeping the number of posts and thus the file size constant. The desired top-level (lowest) resolution partition was determined by considering download time and area covered by a partition. The maximum acceptable download wait for a file was judged to be 30 seconds. On a low bandwidth connection then, such as a dial up 28.8 kilobit per second modem, and allowing for throughput reduction resulting from data control overhead, the maximum file size could not exceed 100 KB. Due to the area of the MBNMS, this results in a 2,000m post-spacing top-level file, with resolution doubling at each step.

C. FILE NAMING CONVENTION

A critical requirement for automated development of a large-scale database is that file names be self-describing in order that both human-readable and machine-readable. Therefore, a file-naming convention was developed that included pertinent attributes of each file in the file's name. The name specifies the location of the file's data, the postspacing, and the data source. Figure 4.1 shows the convention and gives an example file name. File naming is a general requirement for Web-based terrain. This convention likely has wide applicability.

File attributes	Example	Explanation
location latitude	N353610	North 35°36'10"
location longitude	W1232629	West 123°26'29"
north-south minutes	140	140' = grid height
east-west minutes	101	101' = grid width
post spacing, meters	2000	2,000 = post spacing
data survey	SEABEAM	source of data
file type	GRD, WRL, JPG	GRD =text-based grid data file WRL = VRML file JPG = image file (texture)

Example File Name: `N353610.W1232629.140.101.2000.seabeam.wrl`

Figure 4.1. File Naming Convention

D. METADATA

Metadata or "data about data" describe the content, quality, condition, and other characteristics of data. Metadata are used to organize and maintain data, to provide information to data catalogs and clearinghouses, and to aid data transfers. Metadata for each file is included in the header of each file's data contents. Metadata includes

information about filename, post spacing, grid coordinate parameters, and data processes used. A VRML construct developed to encapsulate metadata is explained in Chapter VI.

E. SUMMARY

This chapter provides information about the data used that forms the foundation of the MBNMS Terrain Model. The origin of the raw data is covered, and the processes of gridding and partitioning the data are explained. Also discussed is the reasoning used to determine the grid resolutions. The file-naming convention is described, and an example file name given. Finally, metadata considerations are noted.

V. JAVA PROGRAMS FOR DATAFILE CONVERSION TO VRML

A. INTRODUCTION

Two Java programs were written to produce two types of VRML files that represent the MBNMS as a 3D model. The first program uses gridded data files provided by Moss Landing Marine Laboratories to create the necessary VRML terrain syntax. This Java program also automates the tile-positioning process, whereby tiles are aligned to correspond with their real-world geographic locations. To link together the terrain tiles, a second Java program generates VRML scenes that contain embedded VRML trees which select the level of detail appropriate for the scene relative to the viewer's location.

B. CREATEVRMLTILE PROGRAM: GENERATING INDIVIDUUAL VRML TERRAIN TILES

As noted in Table 4.1, MLML generated a total of 21 gridded data files. For each gridded data file, one VRML tile is required. Thus, 21 VRML tiles needed to be produced. An automated solution to importing the gridded text data and exporting VRML files was produced by writing a Java program. Each VRML tile needed to contain certain cross-linking constructs; these constructs are explained in detail in Chapter VI. Furthermore, VRML tiles need to be offset, or translated, by the distance (in meters) corresponding to the real-world area described by the tile's data. By translating each tile by the correct value, each of the five resolution datasets will appear as a nearly seamless mosaic. For each resolution, a Java class was produced. Appendix C is a CreateVRMLTile Java class that generates terrain tiles using partitions with 1000 meter post spacing. Before running the program, a list is constructed which contains the names of the gridded data files at a given resolution, with one file name on each line. For a

given resolution, the partitions that make up the dataset are essentially simple rectangular pieces of the MBNMS; they can be assembled in the correct order to make a complete representation of the topography at that resolution. In the case of the 500m resolution, there are 16 gridded data files in a 4x4 arrangement that comprise the area of the MBNMS in a two-dimensional (2D) array. The 16 files can be lined up, one after the other, in order of location. For each of the five datasets, with each dataset at a different post-spacing, the CreateVRMLTile program does the following:

1. Read Data File Name

One at a time, names of gridded data files are obtained and operations to produce output based upon that file's data are performed. After getting the name of the first gridded data file in the array, that data file is opened so that its contents may be read.

2. Read Metadata

Next, the program reads the metadata contained in the gridded data file's header. A typical excerpt appears in Figure 5.1.

```
#  *** Filename ***
# N353610.W1232629.070.051.1000.seabeam.wrl
#
# *** Extents Lat/Lon ***
#NW corner
# N36d46m29s
# W123d26m53s
#
#NE corner
# N36d46m30s
# W122d35m47s
#
#SW corner
# N35d36m10s
# W123d26m29s
#
#SE corner
# N35d36m11s
# W122d36m09s
#
# *** Extents UTM ***
#
#NW corner
# 4070000 Northing
#  460000 Easting
#
#NE corner
# 4070000 Northing
#  536000 Easting
#
#SW corner
# 3940000 Northing
#  460000 Easting
#
#SE corner
# 3940000 Northing
#  536000 Easting
#
# *** Post Spacing ***
# 1000 m
#
# *** Original Surfer Header ***
#
# DSAA
# 77 131
# 460000 536000
# 3.94e+006 4.07e+006
# -3811.36 -1449.41
#
```

Figure 5.1. Typical Metadata Excerpt From Gridded Text Data File

3. Read Elevation Data

The program reads the elevation posts which are used to construct a VRML elevation grid.

4. Geographically Position Tile

The program computes, in meters, the necessary distance to translate the VRML tile. This distance is used to geographically position the tile in the MBNMS mosaic. Also computed and written to file is the center coordinate of the tile, to be used later in the second type of Java program, described later in this chapter.

5. Write VRML Syntax

The file-naming convention is observed throughout this process. The program writes VRML syntax to a file whose name corresponds to the gridded data file. A VRML header is produced along with a VRML construct to recreate the metadata of the data file. The program then produces VRML constructs that create a navigation icon, an elevation grid, and local viewpoints that are tied to the navigation icon. The viewpoints are exported to a separate file that is inlined into the top-level scene.

As each individual terrain tile files is completed, the name of the next gridded data file is obtained and the above process repeated until there are no more gridded data files to import.

C. CREATEVRMLTREE PROGRAM: GENERATING LINKING VRML TERRAIN TREES

The VRML construct that performs the switching between resolutions requires a tree structure whereby a parent tile can be exchanged for four children tiles. These trees are then nested to produce the switching architecture. Two tree levels are required for the

overall application, corresponding to the two top-most resolutions, i.e., 2000m and 1000m. Trees do not need to be generated for the highest-resolution tiles, i.e., the 16 500 m tiles. A Java program was developed to automate the process of generating the trees. Appendix D is a CreateVRMLTree Java class that generates four 1,000 m-resolution trees. The works in the following manner:

1. Read Children File Names

For each of parent generated in the tree, the program reads in the names of the files that become the children of that parent and stores those names in an array.

2. Construct Parent and Children Relationship

The program builds the tree structure by extracting from the array the proper children for that tree and computes the parent's name based upon child1's name. The program then reads the center VRML coordinates of the parent tile, which were exported to file as a result of running the tiling program described in section B of this chapter.

3. Write VRML Syntax

Using the information stored from the above procedures, the program then writes the VRML syntax that creates the file that contains the tree that corresponds to the tile for that level of resolution and location. One tree consists of one parent and four children. The process is then repeated until there are no trees left to create for the given resolution. The overall process is repeated again at the next post-spacing resolution.

D. SUMMARY

This chapter describes the process of generating VRML syntax. Two Java programs are introduces that automate the process of generating VRML syntax. The first program reads gridded data files, selects the proper latitude-longitude area, and exports

VRML terrain syntax to create the tile structure of the MBNMS mosaic. The second program creates VRML syntax to create files that contain VRML constructs, which select a level of detail appropriate for the scene relative to the viewer's location.

VI. VRML SCENE DETAILS

A. INTRODUCTION

This chapter describes the VRML constructs contained in the two sets of VRML files that together comprise the MBNMS Terrain Model. The first set of files, the terrain tiles, contain the VRML nodes that create the actual geometry of the shapes in the scene. The second set of VRML files, the terrain trees, contain the VRML nodes that acts as a switch to trade a low-resolution parent tile for a group of higher resolution children tiles.

B. TERRAIN TILES

For a given resolution, many terrain tiles make up the entire MBNMS. Table 4.1 summarizes tile file size and inventory. Terrain tiles are a textual description of a virtual seafloor region. Appendixes E and F are an example terrain tile and scene graph, respectively, for file N353610.W1232629.070.051.1000.seabeam.wrl. Terrain tile constructs are explained below.

1. Metadata

A *Metadata* prototype node has been proposed (Lipkin, 98) which contains pertinent information relating to the contents of a VRML scene. For this project, metadata includes authorship, data origin, geographic coordinates covered by tile, and other relevant tile-specific information. Implemented using VRML's EXTERNPROTO facility, the node permits the Metadata node information to be contained in a string written in Extensible Markup Language (XML) format (XML, 98). Since this exposed field contains XML code in-line as well as the location of an XML document (containing the same metadata) the Metadata node allows easy access to the metadata contents.

As far as possible, the XML keys and key values used correspond to the Dublin Core Metadata Element Set, a 15-element set intended to facilitate discovery of bibliographic information and electronic resources (Weibel, 98). Also considered in determining the key/key value pairs was the Federal Geographic Data Committee Metadata Standard which requires federal agencies to document the data that they produce beginning in 1995 (FGDC, 94).[5] Using these references as a guide, the XML keys for the MBNMS model were developed. The keys (with example key values) are shown in Figure 6.1. These key/key value pairs also appear inlined in the final VRML file (e.g., N353610.W1232629.070.051.1000.seabeam.wrl. in Appendix E) and in the corresponding (XML file N353610.W1232629.070.051.1000.seabeam.xml) in Appendix G). Because of the Metadata node's recent availability, it is not a functioning node in the MBNMS model. However, the syntax for a potential functioning node is provided in each terrain tile along with the actual metadata. Extensive additional work is expected in the funtionality of Metadata nodes, as well as in widespread application of metadata key/key value conventions in terrain data files.

[5] An excellent example of a NOAA implementation of the FGDC standard can be found at *http://www.csc.noaa.gov/metadata/text/seaview_metadata.html*

Key = "Key Value"

Title = "Monterey Bay National Marine Sanctuary Terrain Model
Subject = "VRML Terrain Model"
Pulication_Place = "Monterey, California"
Publication_Date = "1998"
Creator = "Greg Leaver"
Contributor = "Don Brutzman and Ray McClain"
Originator = "Naval Postgraduate School"
Data_Source = "NOAA Seabeam Survey"
Data_Format = "Surfer DSAA Export"
Datum = "NAD83"
Ellipsoid = "GRS 1980"
UTM_Zone = "10"
North_Bounding_Coordinate_UTM = "4200000 N"
South_Bounding_Coordinate_UTM = "3940000 N"
West_Bounding_Coordinate_UTM = "460000 W"
East_Bounding_Coordinate_UTM = "612000 W"
Northwest_Bounding_Coordinate_Latitude = "N37d56m4"
Southwest_Bounding_Coordinate_Latitude = "N35d36m1"
Northeast_Bounding_Coordinate_Latitude = "N37d56m2"
Southeast_Bounding_Coordinate_Latitude = "N35d35m5"
Northwest_Bounding_Coordinate_Longitude = "W123d27m1"
Southwest_Bounding_Coordinate_Longitude = "W123d26m2"
Northeast_Bounding_Coordinate_Longitude = "W121d43m3"
Southeast_Bounding_Coordinate_Longitude = "W121d45m4"
maxElevation = "6.50001m"
minElevation = "-3808.92m"
postSpacing = "2000m"

Figure 6.1. Example XML Keys and Key Values Used in Metadata Node.

2. Positioning

A pair of *Transform* nodes is used to laterally position each tile to its correct geographic location in the MBNMS. Thus all the tiles for a given resolution can be positioned to form a mosaic of the bay. Thus three complete mosaics have been produced, for post-spacing resolutions of 2000, 1000, and 500 meters, respectively. Each

tile file has a single corresponding Transform node referenced to the local origin of the UTM coordinate system, which occurs at the northwest corner of the MBNMS. The topmost linking file has the transform that moves the entire geometry set to the proper UTM coordinates of the northwest-corner of the origin, i.e., 4,200,000 meters Northing, 460,000 meters Easting.

3. Navigation Icons

To enable a user to easily navigate the model a Transform node is defined as a *Navigation Icon.* This node was constructed by inlining and grouping various shapes. *Anchor* nodes are defined within the marker to permit the viewer to select from four different viewpoint orientations from the perspective of that tile: north, south, east, and west. The icon also uses a *LOD* node to switch between high, medium, low, and zero detail shapes depending on the viewer's perspective to the marker. Additionally, the icon provides information about the location it corresponds to by specifying the position in UTM coordinates. An example icon is shown in Figure 6.2. VRML code for the icon appears in file N353610.W1232629.070.051.1000.seabeam.wrl in Appendix E.

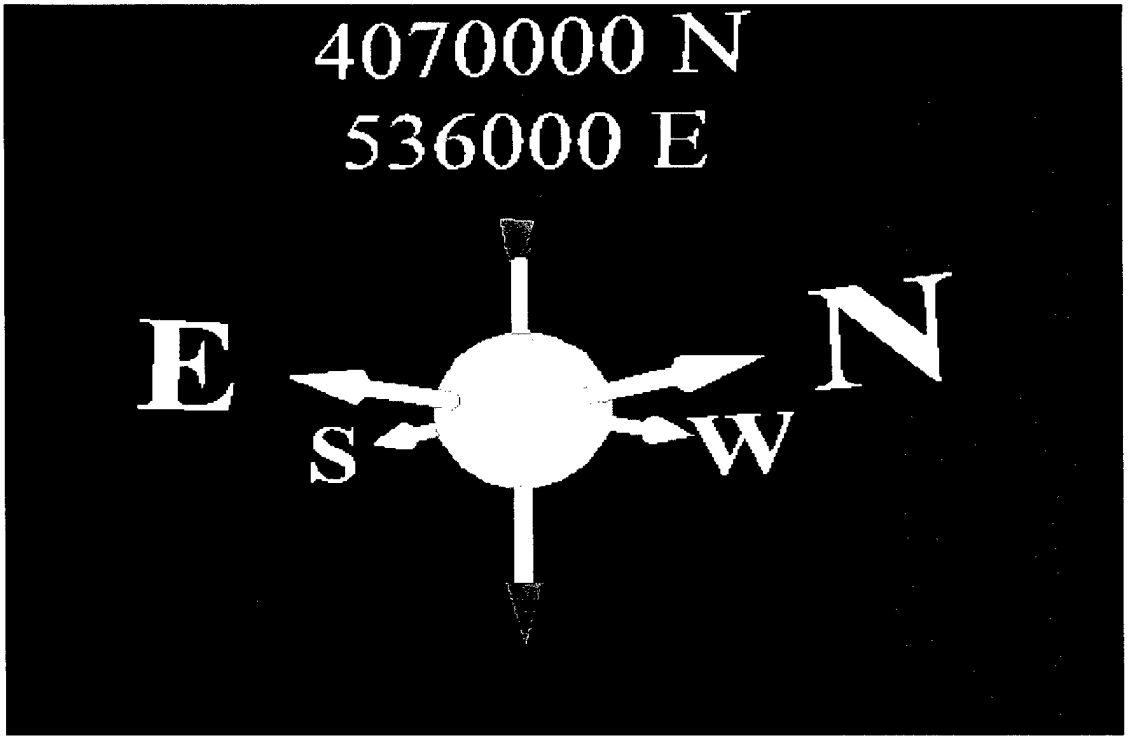


Figure 6.2. A Navigation Icon.

4. Elevation Grids

Each VRML tile has an *ElevationGrid* node, which simulates the topographical features of the geographical region covered by the gridded data file. An example elevation grid is shown in Figure 6.3. Example code appears in file N353610.W1232629.070.051.1000.seabeam.wrl in Appendix E.

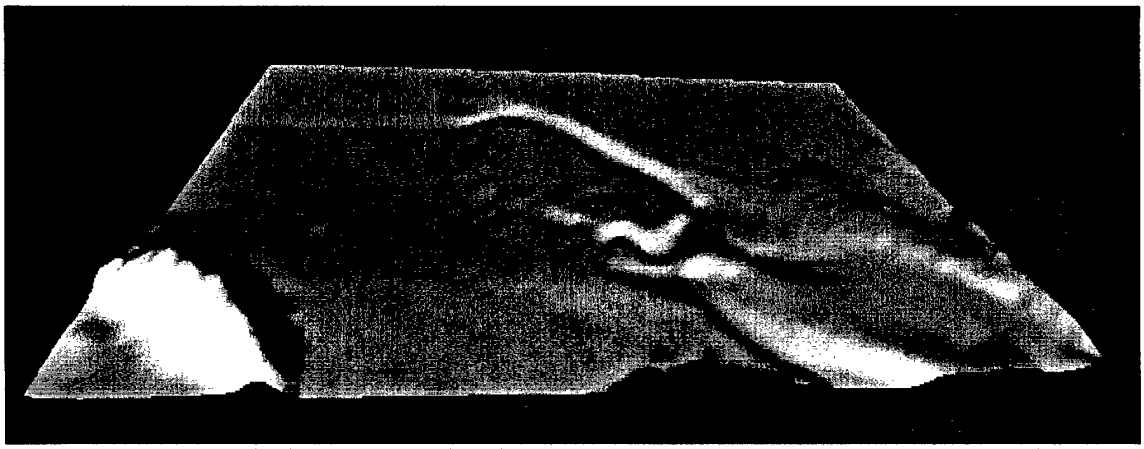


Figure 6.3. A Single Elevation Grid (File N353610.W1232629.070.051.1000.seabeam.wrl)

5. Textures

Each VRML tile has a corresponding JPEG image for a texture. The textures were generated by MLML (McClain, 98) along with the gridded data files. Each image's area corresponds precisely with the area covered by the grids and follows the file-naming convention. Example code appears in file. N353610.W1232629.070.051.1000.seabeam.wrl 1 in Appendix E. These particular textures use a false-coloring technique that maps depth to pixel color. Alternate texture images might easily be substituted. This is an important area for future work.

C. TERRAIN TREES

Terrain trees contain the framework for switching between a low-resolution parent tile and four higher-resolution children tiles. These files do the mechanical work of switching but contain no constructs to create scenery. Appendixes H and I are an example terrain tree and scene graph, respectively, for file N353610.W1232629.070.051.1000.seabeam.wrl. Terrain tile constructs are explained below.

1. Switching

To switch between different tile resolutions, the EXTERNPROTO *QuadLOD* node (Reddy, 98) is employed. A detailed description of the QuadLOD node is beyond the scope of this thesis, but a general overview is provided here. In short, Reddy's node uses a location-to-camera proximity sensor to control the switching between a parent tile and four (quad) children tiles. Additionally, the quad is not fetched until the proximity sensor fires, conserving bandwidth. Also, tiles not within a certain bounding box are not drawn, improving browser-rendering speed. File caching and release has also been added

to improve performance. For these reasons, the QuadLOD node was chosen to form the basis for the QuadSwapTile switching methodology that the MBNMS model depends upon. The values used to control the QuadLOD node's proximity sensor are shown in Table 6.1. These values are implemented by invoking the QuadLOD node as shown in the code excerpt in Figure 6.5, and also appear in Appendix H in file N353610.W1232629.070.051.1000.tree.wrl.

Resolution Transition From Low to High (Low/High; reversible)	Proximity Sensor Box Size (Switch Values)		
	X	Y	Z
2,000m / 1000m	152,000m	100,000m	260,000m
1,000m / 500m	76,000m	50,000m	130,000m

Table 6.1 QuadLOD Node Proximity Sensor Values

```
QuadLOD {
        parentUrl "1000m/Tiles/N353610.W1232629.070.051.1000.seabeam.wrl"
        child1Url "500m/Tiles/N353610.W1232629.035.025.0500.seabeam.wrl"
        child2Url "500m/Tiles/N353613.W1230119.035.025.0500.seabeam.wrl"
        child3Url "500m/Tiles/N361123.W1230120.035.025.0500.seabeam.wrl"
        child4Url "500m/Tiles/N361120.W1232641.035.025.0500.seabeam.wrl"
        parentCenter 38000 0 195000
        parentSize   76000 10000 130000
        switchSize   76000 50000 130000
}
```

Figure 6.4 QuadLOD Excerpt (File N353610.W1232629.070.051.1000.tree.wrl)

2. Viewpoints

Viewpoint nodes are located in the top-level tree. Viewpoints are defined and named to correspond to uniform resource locations (url) specified in Anchor nodes embedded in the Navigation Icons described in the Terrain Tiles section, above. Although four viewpoints correspond to each tile, the syntax for the viewpoints is not located in the tiles or the individual trees. Instead all viewpoint nodes are located in the

top-level 2000m terrain tree file, (i.e., file mbnms.wrl). This is because in order to maintain the switching integrity of the nested trees, the top-level file must always be specified as the url in the browser's netsite window.

D. SUMMARY

This chapter explains the structure of the VRML files that make up the MBNMS Terrain Model. The terrain tiles are gridded adjacent scenes that contain VRML constructs to enable navigation, tile translation, and elevation grid and texture rendering. The terrain trees provide the mechanism needed to switch between levels of detail. The top-level tree contains viewpoints, while the Navigation Icons point to those viewpoints.

VII. EXPERIMENTAL RESULTS

A. INTRODUCTION

This chapter presents results obtained in the implementation of the terrain model. It takes a look at the structure of the database and the size of the datasets. Next, the performance of the model is measured by timing how long it takes to render resolutions on two different machines. A section on user access and navigation provides the web address and user information on how the model can be accessed and navigated. The final section discusses how the model is georeferenced with the UTM coordinate system and shows how a user can add content to the MBNMS scene by taking advantage of this georeferencing.

B. MONTEREY BAY TERRAIN MODEL DATABASE

The directory structure of the model is illustrated in Figure 7.1.

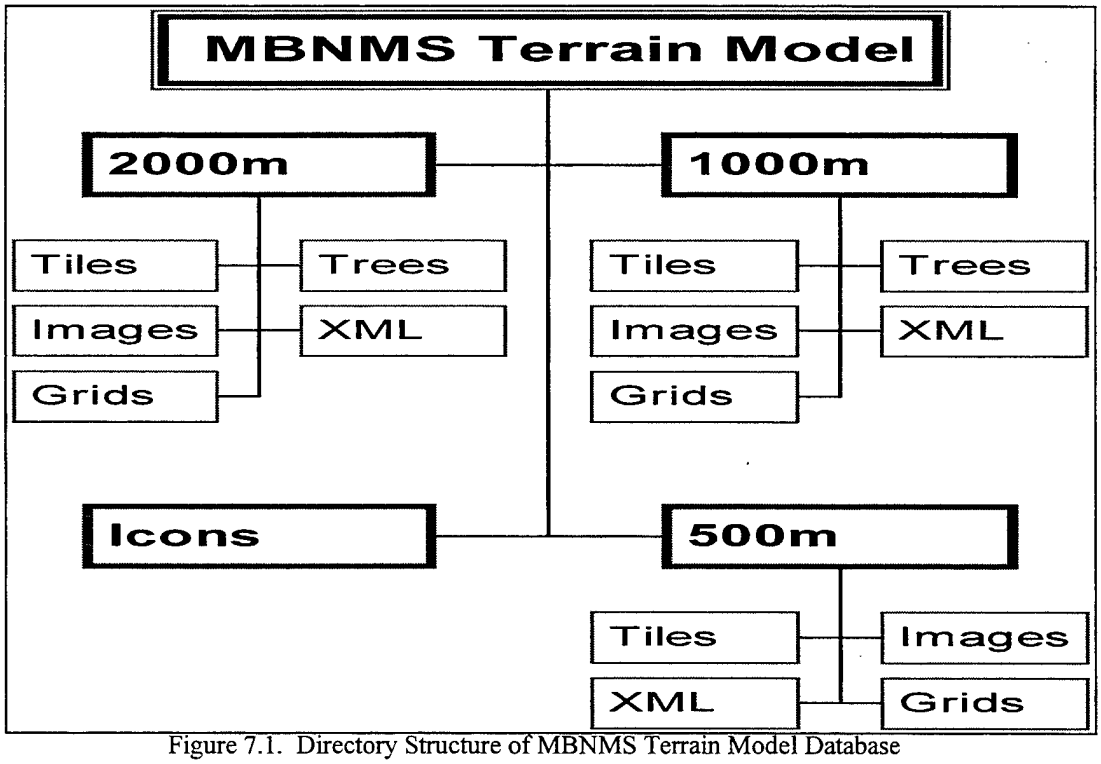


Figure 7.1. Directory Structure of MBNMS Terrain Model Database

Table 7.1 gives the detailed characteristics of terrain tiles, by resolution (subdirectory) while Table 7.2 gives the detailed characteristics for the terrain trees, again by resolution. The directory structure of the images and XML datasets mirror that of the terrain tiles dataset.

Resolution (Post Spacing)	Number of Tiles	Tile Area	Total Area (all tiles)	Range of Individual File Sizes (Tiles)	Cumulative File Size at This Resolution (Tiles)
2000m	1	3,952,000 km^2	3,952,000 km^2	60 KB	60 KB
1000m	4	9,880,000 km^2	3,952,000 km^2	51-67 KB	239 KB
500m	16	2,470,000 km^2	3,952,000 km^2	51-67KB	959 KB
			Total Cumulative Size of Tile Datasets		1258 KB

Table 7.1. Terrain Tile Dataset Characteristics

Resolution (Post Spacing)	Trees	Parent Tiles	Children Tiles	Range of Individual File Sizes (Trees)	Cumulative File Size at This Resolution (Trees)
2000m	1	1	4	17 KB	17 KB
1000m	4	4	16	2 KB	8 KB
			Total Cumulative Size of Tree Datasets		25 KB

Table 7.2. Terrain Tree Dataset Characteristics

C. PERFORMANCE RESULTS AND USABILITY TESTING

Currently, the MBNMS Terrain Model can be accessed by any VRML browser. However, going past the top-level low-resolution (2000m post-spacing) entry file is only possible using a PC or Macintosh. This is because the model uses *Script* nodes that interface with Java class files. Currently, only VRML browsers designed for PCs follow

the VRML 97 specification as it pertains to Java in Script nodes (Brutzman, 98). Therefore, the model currently does not show higher-resolution tiles with Cosmo Player under IRIX (SGI) because it does not support Java in the Script node. This limitation is expected to be temporary since all VRML used in the MBNMS model is compliant with the VRML 97 specification (VRML 97, 97)

1. Performance Aids

a. Vertical Exaggeration.

Even though VRML renders geometry in 3D, it helps to exaggerate the depth on a 2D screen. Increased vertical relief can accomplish this. The MBNMS Terrain Model has a vertical exaggeration of one. Figure 7.2 shows how vertical exaggeration can be applied, albeit at a loss of scale and registration with any other associated object geometry. Figures 7.3 and 7.4 show example differences in vertical exaggeration.

```
Transform {
        scale 1 3 1 # applies non-uniform scaling (3:1) along vertical-axis (depth)
        children Inline {url " N353610.W1232629.070.051.1000.seabeam.wrl" }
}
```

Figure 7.2 Applying Vertical Exaggeration

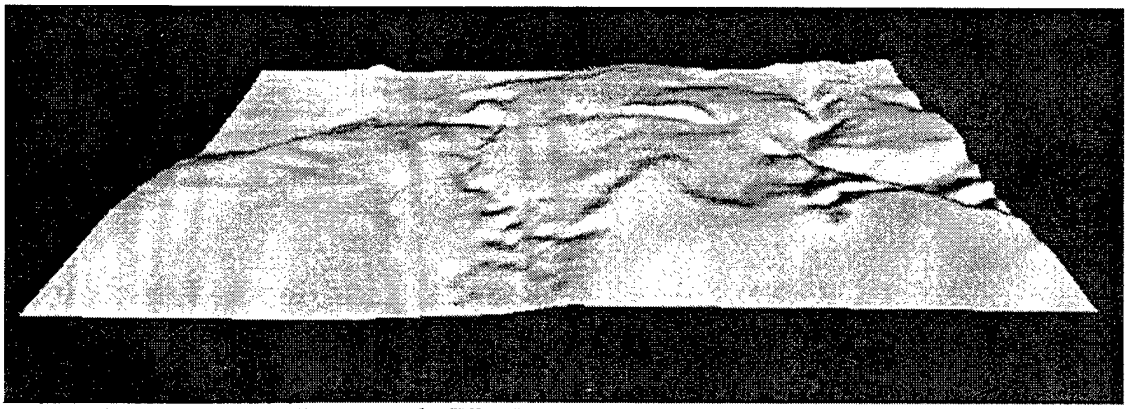


Figure 7.3. 3:1 Scaling Example (File N353611.W1223609.070.051.1000.seabeam.wrl)

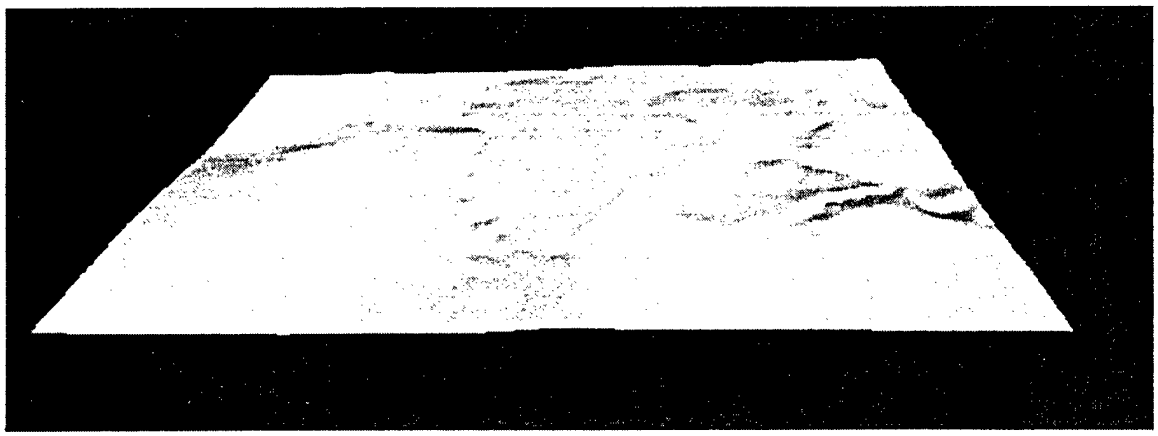


Figure 7.4. 1:1 Scaling Example (File N353611.W1223609.070.051.1000.seabeam.wrl)

b. Reducing File Size by Rounding

File size of terrain tiles is reduced by 30 - 40 % by using integers for bathymetric depth values rather that floating point numbers. In an oceanic environment with depths in thousands of meters, 0.5 meters of error due to integer rounding is an acceptable figure considering the payback in download and rendering speed that results from substantially smaller files. Since maximum resolution of the underlying smoothed dataset ranges from 5m to 10m, this is a safe assumption.

2. Performance Results

The MBNMS contains three levels of resolutions, 2000m, 1000m, and 500m. Table 7.3 shows the rendering performance on two different machines, with time as the performance measure. Time is measured from the moment a proximity sensor is tripped from one resolution (to load the next resolution) until the next resolution is drawn on the screen. Measurements are for the exchange of one parent tile for four children tiles and are not cumulative. On slower processors, or those without 3D-graphics hardware acceleration, the model renders reasonably quickly at the 2000m and 1000m resolutions.

But on these platforms the 500m resolution tiles are slow to render. Performance was relatively unaffected by whether or the copy was on a local machine or fetched over a network. Much further user testing and optimization is possible. Future implementation of SwapTile and QuadTile mechanisms as described in Chapter IV can likely provide good speedups across all platforms. Nevertheless, we believe that the current approach already provides excellent scalability, since high-resolution files need only be retrieved in close proximity to the viewer.

Resolution From/To (reversible)	**Rendering Time in Seconds**		
	Local Copy	Network Copy	
	Machine 1 P 133 MHz	Machine 1* P 133 MHz	Machine 2# P 166 MHz
Entry Level Load - 2000m	40s	73s	18s
2,000m / 1000m	52s	186s	87s
1,000m / 500m	65s	143s	112s

Table 7.3. Rendering Time for Resolutions

D. USER ACCESS AND NAVIGATION

The MBNMS Terrain Model is available over the Web at *http://www.stl.nps.navy.mil/~auv/leaver/MBNMSterrain/2000m/Trees/mbnms.wrl.* The model is compatible with Cosmo Player (minimum version 2.1). The entry-level viewpoint presents the top-level scene shown in Figure 7.5.

* 28.8 Kilobits per second Modem Connection
# Dedicated Internet Connection

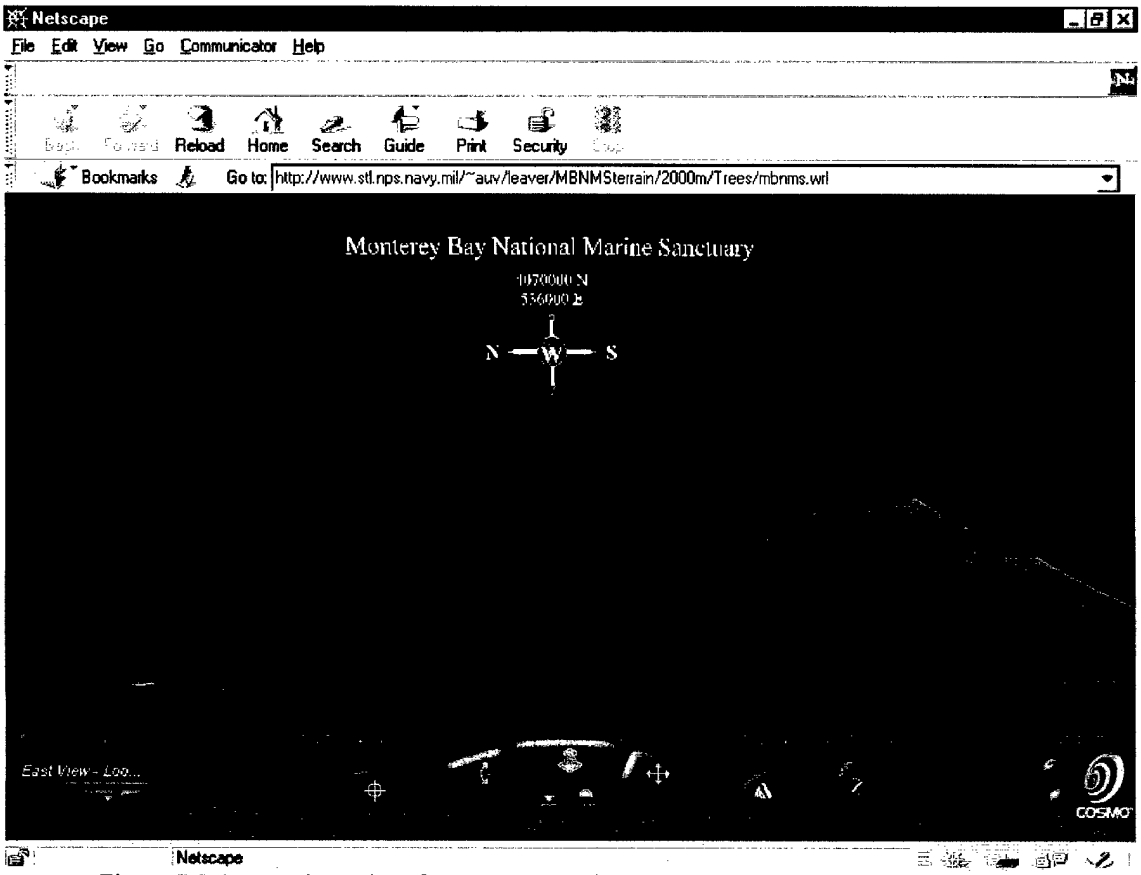


Figure 7.5. Entry Viewpoint of MBNMS Terrain Model

A navigation icon is positioned at the center of the tile in the screen foreground. Navigation icons are switched in and out together with corresponding terrain tiles. Each tile has a unique navigation icon that provides four relative viewpoints. The initial top-level scene is at the widest possible resolution (2000 m). The viewer can determine this by positioning the mouse over any of the navigation icon's four compass headings. This causes the current level of resolution (that the navigation icon is at) to be displayed I at the bottom of the browser's window. By clicking on an icon direction label a viewpoint can be selected. Selecting the viewpoint causes the viewers position to change to the direction of the compass heading chosen, at a perspective just above the navigation icon. Navigation icons are positioned at a tile's center on the boundary of the proximity sensor that controls the switch between resolutions. That is, if the viewer flys beneath or drops

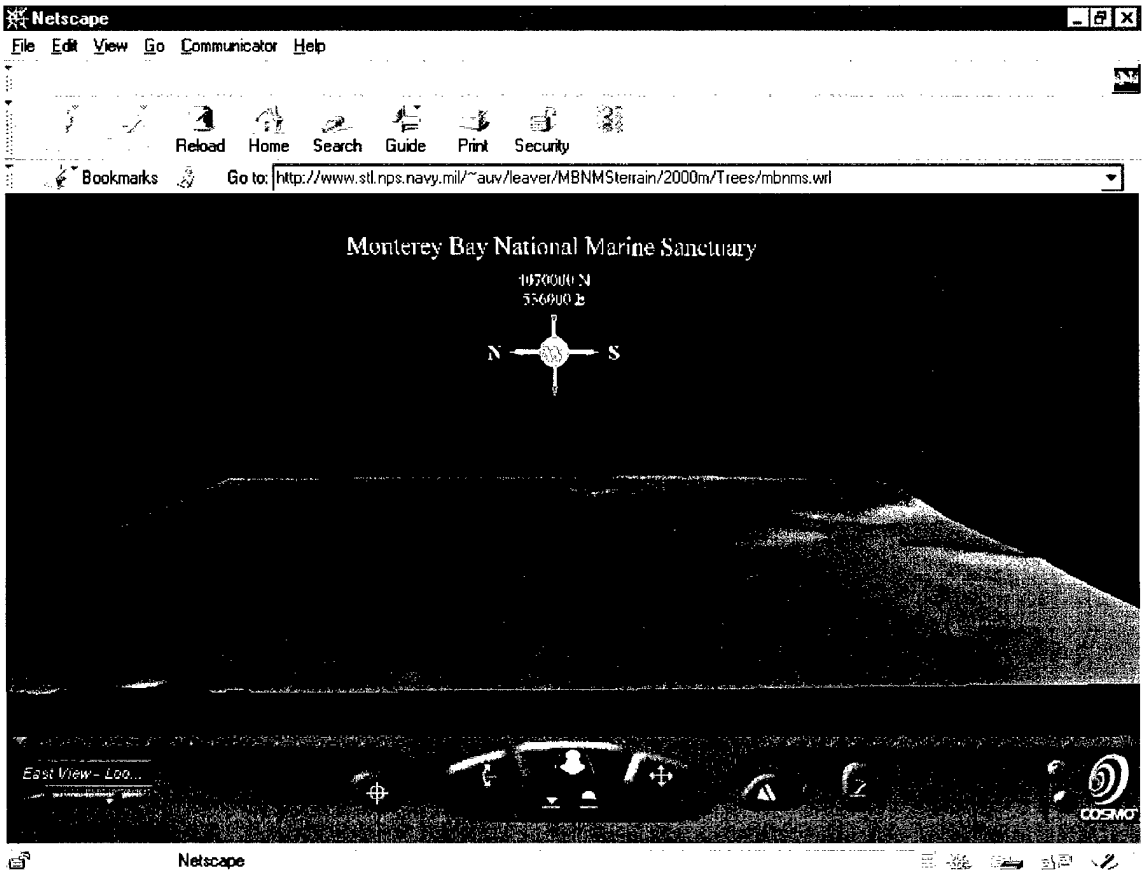


Figure 7.5. Entry Viewpoint of MBNMS Terrain Model

A navigation icon is positioned at the center of the tile in the screen foreground. Navigation icons are switched in and out together with corresponding terrain tiles. Each tile has a unique navigation icon that provides four relative viewpoints. The initial top-level scene is at the widest possible resolution (2000 m). The viewer can determine this by positioning the mouse over any of the navigation icon's four compass headings. This causes the current level of resolution (that the navigation icon is at) to be displayed I at the bottom of the browser's window. By clicking on an icon direction label a viewpoint can be selected. Selecting the viewpoint causes the viewers position to change to the direction of the compass heading chosen, at a perspective just above the navigation icon. Navigation icons are positioned at a tile's center on the boundary of the proximity sensor that controls the switch between resolutions. That is, if the viewer flys beneath or drops

below a navigation icon, the next resolution for that tile will be switched in. Thus, a low-resolution tile (and the tile's navigation icon) is replaced with four higher resolution tiles (and their corresponding navigation icons). An icon can be made to disappear by clicking on its sphere. Icons which have been made to disappear can be restored by zooming out to a lower-resolution level and then zooming back in to reload the higher-resolution level. The icons and switching occurs at the values shown in Table 7.3.

Resolution Switching Event (Reversible)	Switch Value (Icon Elevation)
2000m to 1000m	50,000m
1000m to 500m	25,000m

Table 7. 4 Tile Switching Values (depth)

E. EXAMPLE INTEGRATION OF CONTENT

1. Georeferencing

The MBNMS Terrain Model's coordinate system is georeferenced to the UTM coordinate system. The model is built with the VRML coordinate system origin (the point 0,0,0 of the root node) corresponding to the baseline position of Zone 10 of the UTM coordinate system, i.e., 0 m North, 0 m East of Zone 10. The northwest corner of the real MBNMS is located at 4,200,000 m north and 460,000 m east (in UTM, 4,200,000N, 460,000E) of the zone 10 baseline. In VRML, the Terrain Model is designed so that North is in the **positive x** direction, and East is in the **positive z** direction. So, the northwest corner of the Terrain Model is located 4,200,000 units in the positive x direction, and 460,000 units in the positive z direction. Thus, in the x-z plane, the coordinates of the Terrain Model correspond to the real life sanctuary in terms of the

UTM coordinate system. The y-axis represents depth. Figure 7. 6 represents these relationships.

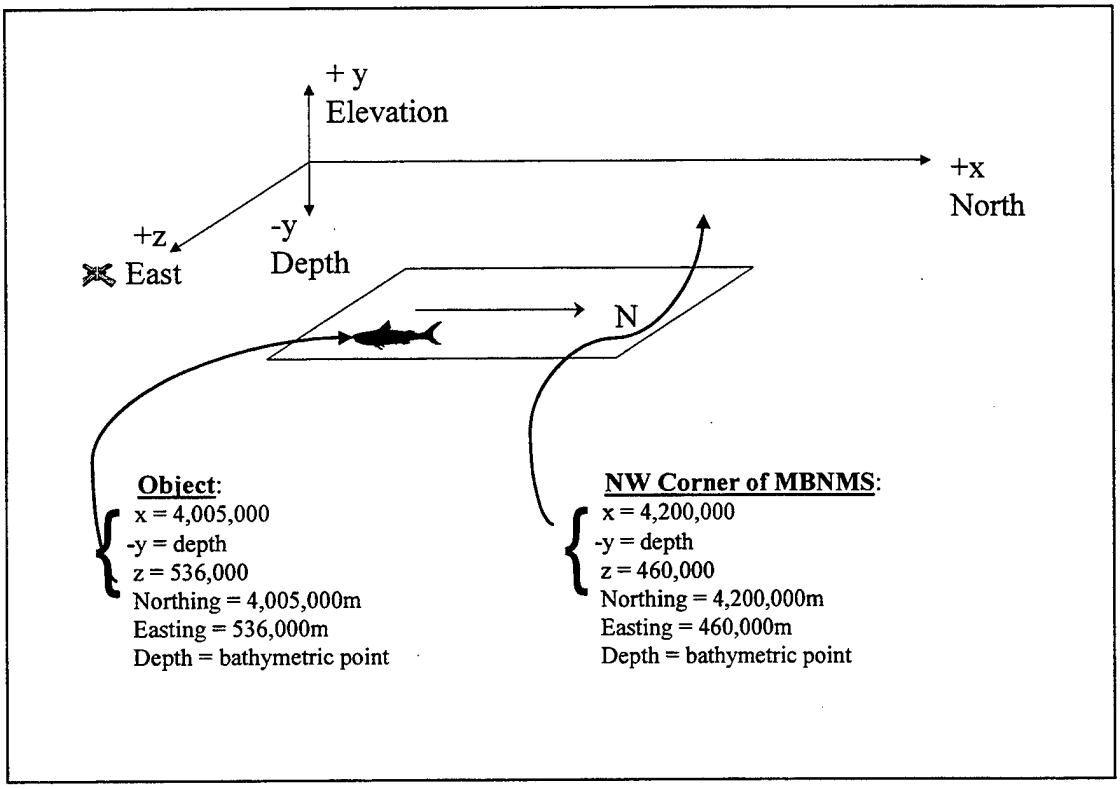


Figure 7.6. Georeferencing VRML and UTM Coordinate Systems

It is worth noting that this two-stage approach to georeferencing was further necessary because VRML 97 does not support double floating-point precision. Establishing two-step floating-point coordinate conventions, or providing double-precision support are important areas of effort for the GeoVRML Working Group.

2. Adding Content

Since the MBNMS Terrain Model is georeferenced with the UTM coordinate system, adding content to a VRML scene in context with its proper location in the real MBNMS requires just a few simple VRML statements. As Figure 7.7 shows, content can be added in context with the MBNMS that might be at the location of 4005000 Northing and 536000 Easting (i.e., N36d11m20s latitude, W122d35m58s longitude).

F. SUMMARY

This chapter describes the detailed VRML construction of the MBNMS database and also examines the structure and the size of the datasets. A section on performance measures how long it takes for the model to render on two different machines. Also, a section on user access and navigation provides the web address to enable a user access the model and information on how the model can be navigated. Finally, by georeferencing the model with the UTM coordinate system, an example is provided to show the user how content to the MBNMS scene can be added.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. RESEARCH CONCLUSIONS

The VRML terrain model of the Monterey Bay National Marine Sanctuary consists of numerous terrain tiles, terrain trees, and jpg images to support three levels of topographic resolution. Different VRML mechanisms for switching between resolutions have been considered, with the QuadLOD selected as the primary mechanism of choice. Relevant findings on issues regarding VRML creation, viewpoints, rendering, and georeferencing with content are amplified below.

1. Generating VRML Syntax

To build the MBNMS model, preprocessed bathymetric data files are imported by a Java program which positions the terrain tiles and terrain trees and exports VRML syntax. A prerequisite effort was access to a large GIS database engine (MLML, 98). This approach proved to be an effective method of creating VRML files for a large-scale virtual environment.

2. Viewpoints

Extensive experimental efforts to embed native VRML viewpoints in the individual tiles failed, for two reasons: self-describing viewpoint names are too long, and viewpoints continued to accumulate far past any usable number.

The current model uses a navigation icon to select viewpoints and to mark locations in the VRML world that correspond to real-life locations (e.g., the center of each tile). One of the benefits of using navigation icons is that they can be inlined with the tile geometry. This allows them to be switched in and out with terrain tiles; automatically culling unneeded icons from view along with the icon's corresponding

terrain tile. Another benefit is that four viewpoints are attached to the icon to enable easy viewer navigation of the model. The viewpoint nodes, while defined in the top-level scene, are pointed to from navigation icons that are located in the terrain tiles.

The viewpoint nodes in the top-level scene contain empty description fields. If the description field is empty, viewpoints aren't listed in the browser window. Ordinarily, viewpoint descriptions are listed, but with the QuadLOD switching scheme utilized by the model, four viewpoints would be listed each time a tile is switched in. Unfortunately, the viewpoint list is not culled when tiles are dropped from view or switched out, causing the viewpoint menu to be quickly overloaded. The navigation icon method solves this problem since the empty description field in the viewpoint node prevents the viewpoints from ever appearing in the viewpoint menu. It also frees up the viewpoint menu for an author's content-related use.

3. Rendering

The MBNMS contains three levels of resolutions: 2000m, 1000m,and 500. On slower processors, or those without hardware acceleration, the model renders reasonably quickly at the 2,000m and 1,000m resolutions (one or two frames per second). But, on these platforms, the 500m resolution is slow to render. This does not seem to be related to file size of the terrain tiles or file download delays, but instead seems to be a result of too many polygons in the scene. The QuadLOD node does cull tiles that are not visible, so it seems the problem lies with too-high resolution tiles in the scene, given the area covered by the scene. In other words, there are too few resolution steps nested in the tree hierarchy. This probably might be further diagnosed and solved by starting with a lower-

resolution top-level scene than the 2000m spacing used, such as 4000 m spacing. Resolution can then be increased again by a factor of two each time.

4. Georeferencing and Content

The MBNMS Terrain Model, built in the VRML coordinate system, is georeferenced to the real sanctuary when mapped in a UTM system. By aligning the model to the UTM coordinate system, users are able to place content in context with a terrain environment. A few simple VRML statements added to the user's scenes are all that's necessary to integrate geometry. However, this content is most likely at a depth encapsulated by a high-resolution proximity sensor. As a result, if a user attempts to get close to the object to view it in detail, the sensor will trigger, and as mentioned above, this high resolution scenery is slow to render on some machines, making accessibility to the object problematic.

B. RECOMMENDATIONS FOR FUTURE MBNMS TERRAIN MODEL WORK

1. Normal Vectors to Eliminate Tile Seams

When a quad of higher resolution child tiles replaces a lower resolution parent tile seams between the quad of tiles are sometimes noticeable. Although this is expected when tiles of different resolutions abut one another, the seams occur even within the quad. These tiles - and neighboring tiles of the same resolution - are in fact coplanar because they were designed to have the same elevation points at the seams. The problem is that *normals* [6] (normal vectors) for the two adjoining tile's edges are computed differently and so the edges are shaded differently. This occurs because while the edge

[6] Unless explicity specified, normals are computed by the browser. Normals are like arrows pointing straight out from a face or an edge, and determine shading.

for one tile might face the viewer, the edge for its neighbor might be facing away from the viewer. Since each tile has no information about the other, the normals are computed independently. To resolve this inconsistency, normals can be preprocessed and explicitly included in the node's geometry for a scene. Explicitly specifying normals might also aid in rendering speed, as it relieves the browser from the computational effort during initial rendering. However, it will increase download time. Note that normals also have a visible effect if texture imagery is included. A possible optimization for the next-generation VRML specification is to allow definition solely for normals of the edge vertices, allowing browsers to calculate internal normals.

2. Modified QuadLOD Node

The QuadLOD Node is a remarkable node - an efficient algorithm for switching tiles that the MBNMS Terrain Model must have for proper operation. However, one drawback from employing the current node is that when a viewer exits and re-enters the node's proximity sensor box, the node reloads tiles already in memory. This causes delays in rendering and another fetch for four tiles - whether they are local or over the network. Browsers don't seem to be smart enough to yet know the difference, so a solution might be a smarter QuadLOD node that turns off its proximity sensor when the viewer exits the sensor's box. Such an approach could be coded using static (persistent) classes. This would prevent the quad from being reloaded, and would allow a developer to include descriptive viewpoints with the tiles that would not be redundantly loaded and cumulatively bound to the scene. A separate drawback to using the QuadLOD node is that it binds all geometry to the top-level node. For example, to include content in the MBNMS, the content must be inlined or nested with the file containing the top-level

QuadLOD node. This author was unsuccessful in finding a way to do the reverse - i.e., inlining the MBNMS file containing the QuadLOD node with the content. This is a high priority shortcoming which needs correction.

3. Navigation Icons to Control Other Transitions

This thesis discusses three types of switching. The first is the QuadSwapTile method implemented by the QuadLOD node to perform switches automatically. The other two (SwapTile and QuadTile) might be good ways for a user to decompose the combined functionality of the QuadLOD node by allowing a user to either break a parent tile into four children, or to simply trade a low resolution tile for a high resolution tile. If the user might control transitions between these two alternatives, perhaps by using a control panel attached to the navigation icon that was brought in and out with the tile, the user might navigate and transition more easily to the desired resolution. It would also be more efficient since the QuadTile switch might let a user continually segment a low-resolution tile until the destination is approached and the desired locale needs to be "blown up" to a higher-resolution tile. At this time a SwapTile transition could complete the transitions to high-resolution local small-area tiles, saving all the download and rendering overhead that comes with the middle resolutions in the nested hierarchy approach. Either or both transition triggers might be attached sensors or anchors on the navigation icon already available for user selection. A triggered QuadLOD node could be used for the QuadTile transition, while the *VisibliiyInline* node (Reddy, 98) could might be employed to switch tiles in the SwapTile transition.

4. Return of the Navigation Icons

Currently, a viewer can cause navigation icons to disappear by clicking on a certain part of them. They disappear by scaling to near zero. The problem with this approach is that the viewer can't get them back unless the tile that contains them is reloaded. To offer a means of retrieving them, it might be better instead for them to become nearly transparent upon user selection rather than to scale down, so that the icons are still visible (barely) in the scene. The problem with putting a control panel elsewhere (out of the immediate scene) is that the current scene and its geometry will be disrupted as QuadLOD node proximity sensors are tripped and new geometry is loaded when moving to the viewpoint.

5. Other MBNMS Model Future Work

Because the maximum resolution of the underlying smoothed dataset ranges from five meters to 10m, it is possible to construct higher-resolution gridded datasets than the 2000m, 1000m, and 500m datasets generated for this project. Future work includes generating higher-resolution datasets from which terrain tiles with post-spacings of 250m and 125m tiles can be created. Even higher resolution terrain tiles might be developed as well.

Additional work is also needed to provide functionality for the Metadata node as applied in the MBNMS Terrain Model and to improve metadata interoperability.

C. OTHER LSVE FUTURE WORK

With the amount of digital elevation data available it may soon be possible to map the United States as a VRML LSVE. Future work includes an automated mapping of the entire DEM database to produce a VRML model of the continental United States.

Another item ripe for future efforts is the development or evaluation of tools for converting latitude/longitude coordinates into UTM. Still more possibilities exist in developing conventions for texture mapping, especially autogenerated textures. Finally, future work includes establishing SEDRIS interoperability with VRML terrain models. We forsee the day when georeferenced terrain is an expected context for all real-world 3D scenes on the Web.

APPENDIX A: SCRIPTS USED TO GRID DATA SETS

Surfer Script 1:

```
' ************************************
' * makegrid.bas                     *
' * Grids a bathymetric data set     *
' * Runs under the Surfer environment *
' * Programmer: Ray McClain          *
' * Created: 6/98                    *
' ************************************


' ***** Initialize input and output data files *****
DataFile$=InputBox$("Enter the data file name","Make
Grid","C:\public\vrml\batch")
OutPath$="c:\public\vrml\batch\oututm.grd"

' ***** Initialize Windows app variables *****
Set Surf=CreateObject("Surfer.App")


' ***** Initialize process variables *****
Method=1

' ***** Grid the data *****
if
Surf.GridData(DataFile$,GridMethod=1,xSize=250,ySize=250,Out
Grid=OutPath$,OutFmt=2)=0 then end
```

Surfer Script 2:

```
' ************************************
' * plotgrid.bas                     *
' * Extracts tiles from a grid file  *
' * Runs under the Surfer environment *
' * Programmer: Ray McClain          *
' * Created: 7/98                    *
' ************************************

print
print


' ***** Initialize the external files *****
InDataFile$="C:\public\vrml\rev250.grd"
DataFile$=InputBox$("Enter the data file name","Ray's
Test","C:\public\vrml\batch")


' ***** Initialize the Windows system variables *****
Set Surf=CreateObject("Surfer.App")


' ****** Initialize the loop variables *****
FirstRow=1
LastRow=131
fn = 1


' ***** Extract tiles by rows and columns *****
for rows=1 to 8

FirstCol=1
LastCol=77

for cols= 1 to 8


' ***** Create tile filename *****
fn$=ltrim$(str$(fn))
OutPath$=DataFile$+"\"+fn$+"ray250"

print OutPath$
print "FirstCol = ", FirstCol
print "LastCol = ", LastCol
print "FirstRow = ", FirstRow
print "LastRow = ", LastRow


' ***** Extract the tile *****
```

```
if
Surf.GridExtract(InGrid=InDataFile$,r1=FirstRow,c1=FirstCol,
r2=LastRow,c2=LastCol,OutGrid=OutPath$,OutFmt=2)=0 then end

FirstCol=LastCol
LastCol=LastCol+76

fn = fn +1
next

FirstRow=LastRow
LastRow=LastRow+130

next
```

APPENDIX B: SCRIPT USED TO PARTITION DATA SETS

TNTMips Script:

```
# ****************************************
# * ll2utm.sml                           *
# * Converts bathymetry from Lat/Lon to UTM *
# * Runs under TNTMips SML environment   *
# * Programmer: Ray McClain              *
# * Created: 5/98                        *
# ****************************************

clear()


# ***** Open external files *****
f = fopen("c:/public/vrml/sbll.asc","r")
g = fopen("c:/public/vrml/sbutm.dat","w")


# ***** Select georeference parameters for source data *****
print("Choose the source parameters.")
source = GeorefGetParms()

print()
print()


# ***** Select georeference parameters for output data *****
print("Choose the destination parameters.")
dest = GeorefGetParms()

print();print()


# ***** Initialize loop variables *****
a = 0
totalcount = 1
count1 = 1


# ***** Process until end-of-file *****
while(a==0) begin


# ***** Read input data file *****
depth = fgetnum(f)
lat= fgetnum(f)
lon= fgetnum(f)
```

```
# ***** Convert Lat/Lon to UTM
GeorefTrans(source,lon,lat,dest,east,north)


# ***** Write UTM values to output file *****
fprintf(g,"%10.6f %10.6f %7.2f\n",north,east,depth)


# ***** Update the process status bar *****
count1 = count1+1
if(count1 == 10000) then begin
printf(".")
count1 = 1
end  #if(count1 == 50


# ***** Test for end-of-file *****
a = feof(f)

totalcount = totalcount +1
end  #while(a==0


# ***** Print processing summary *****
print();print()
print("Total points converted = ",totalcount)
print();print()
print("***** Conversion Process Completed *****")

# ***** Close external files *****
close(f)
                              close(g)
```

APPENDIX C: CREATEVRMLTILE JAVA PROGRAM

```
//
**************************************************************************
//
// Script Name: Create1000mVRMLTile.java
//
// Description: Create a VRML terrain tile at 1000m post-spacing
resolution
//
// Author:      R. Greg Leaver
// Advisor:     Don Brutzman
//
// Revised:     1 September 98
//
// Notes:       Need one class for each post-spacing
//
//
**************************************************************************
//
**************************************************************************


import java.io.*;
import java.math.*;
public class create1000mVRMLtile {
  public static void main (String argv [ ]) throws IOException {
      FileInputStream dataSetFile = new FileInputStream
("1000mGridSet.txt");
      StreamTokenizer dataSetTokens = new StreamTokenizer
(dataSetFile);
      String gridName;

// Create output stream to write viewpoints to file;
      FileOutputStream viewpointOutputFile = new FileOutputStream
("1000mViewpoints.txt");
      PrintStream viewpointOutput = new PrintStream
(viewpointOutputFile);

// Create output stream to write center of tile values to file;
      FileOutputStream vrmlTreeOutputFile = new FileOutputStream
("1000mParentCenters.txt");
      PrintStream vrmlTreeOutput = new PrintStream
(vrmlTreeOutputFile);

// Set postspacing and number of tiles:
      int postSpacing = 1000;
      int numberTiles = 4;

// Determine tile grid side for use in translation field of Transform
node:
      int sideTiles = (int) Math.sqrt (numberTiles);
```

```
      dataSetTokens.quoteChar((int) '"');

// Initialize translation and dimension values:
      int lastzDimension [ ] = new int [sideTiles];
      int lastzTransform [ ] = new int [sideTiles];
      for (int i = 0; i <= (sideTiles-1); i++) {
            lastzDimension [i] = 1;
            lastzTransform [i] = 0;
      }
      int lastxDimension = 1;
      int lastxTransform = 0;
      int xCounter, zCounter;
      int xDimension = 0;
      int xTransform = 0;
      int zTransform = 0;
      int zDimension = 0;

// Loop for setting z translation value:
      for (zCounter = 1; zCounter <= sideTiles; ++ zCounter) {

//  Loop for setting x translation value:
      for (xCounter = 1; xCounter <= sideTiles; ++ xCounter) {
      dataSetTokens.nextToken ( );

// Get data file names one at a time:
      gridName = dataSetTokens.sval;

// Pass data file name to input stream:
      FileInputStream dataInputFile = new FileInputStream (gridName);
      StreamTokenizer dataInputTokens = new StreamTokenizer
(dataInputFile);

// Variables for data input:
      double minDepth, maxDepth, lowerUTMN, upperUTMN, leftUTME,
rightUTME,
      centerUTMN, centerUTME;
      String fileName, viewpointName,
      uleftLong, uleftLat, urightLong, urightLat,
      lleftLong, lleftLat, lrightLong, lrightLat,
      uleftUTMN, uleftUTME, urightUTMN, urightUTME,
      lleftUTMN, lleftUTME, lrightUTMN, lrightUTME,
      northing, easting;


// Set parser flag to # and begin parsing data file:
      dataInputTokens.quoteChar((int) '#');
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      fileName = (String)dataInputTokens.sval.substring(1,42);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
```

```
      dataInputTokens.nextToken( ); uleftLat =
(String)dataInputTokens.sval.substring(1,9);
      dataInputTokens.nextToken( ); uleftLong =
(String)dataInputTokens.sval.substring(1,10);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); urightLat =
(String)dataInputTokens.sval.substring(1,9);
      dataInputTokens.nextToken( ); urightLong =
(String)dataInputTokens.sval.substring(1,10);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); lleftLat =
(String)dataInputTokens.sval.substring(1,9);
      dataInputTokens.nextToken( ); lleftLong =
(String)dataInputTokens.sval.substring(1,10);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); lrightLat =
(String)dataInputTokens.sval.substring(1,9);
      dataInputTokens.nextToken( ); lrightLong =
(String)dataInputTokens.sval.substring(1,10);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); uleftUTMN =
(String)dataInputTokens.sval.substring(1,8);
      dataInputTokens.nextToken( ); uleftUTME =
(String)dataInputTokens.sval.substring(2,8);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); urightUTMN =
(String)dataInputTokens.sval.substring(1,8);
      dataInputTokens.nextToken( ); urightUTME =
(String)dataInputTokens.sval.substring(2,8);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); lleftUTMN =
(String)dataInputTokens.sval.substring(1,8);
      dataInputTokens.nextToken( ); lleftUTME =
(String)dataInputTokens.sval.substring(2,8);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); lrightUTMN =
(String)dataInputTokens.sval.substring(1,8);
      dataInputTokens.nextToken( ); lrightUTME =
(String)dataInputTokens.sval.substring(2,8);
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
```

```
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );

// Unset parser flag:
      dataInputTokens.ordinaryChar((int) '#');
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); xDimension =
(int)dataInputTokens.nval;
      dataInputTokens.nextToken( ); zDimension =
(int)dataInputTokens.nval;
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); leftUTME = dataInputTokens.nval;
      dataInputTokens.nextToken( ); rightUTME = dataInputTokens.nval;
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); upperUTMN = dataInputTokens.nval;
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); lowerUTMN = dataInputTokens.nval;
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( );
      dataInputTokens.nextToken( ); minDepth = dataInputTokens.nval;
      dataInputTokens.nextToken( ); maxDepth = dataInputTokens.nval;
      dataInputTokens.nextToken( );

// Compute center UTM values:
      centerUTMN = (1000000)*((upperUTMN + lowerUTMN)/2);
      centerUTME = (leftUTME + rightUTME)/2;
      northing = String.valueOf(centerUTMN).substring(0,7);
      easting = String.valueOf(centerUTME).substring(0,6);

// Read data and construct initial array from Surfer program export:
      int i, j;
      int vrmlArray [ ] [ ] = new int [zDimension+1] [xDimension+1];
      for (i = 0; i <= (zDimension -1); i++) {
            for (j = 0; j <= (xDimension-1); j++) {
                  dataInputTokens.nextToken( );
                  vrmlArray [i] [j] = (int)dataInputTokens.nval;
            }
      }

// Construct values to geographically translate the tile:
      xTransform = postSpacing * (lastxDimension - 1 ) +
lastxTransform;
      zTransform = postSpacing * (lastzDimension [xCounter - 1] -1 ) +
lastzTransform [xCounter - 1];
      dataInputFile.close( );

// Compute center of tile for use in creating tree and export to file:
      int xtreeTranslation = xTransform + (postSpacing*(xDimension-
1)/2);
```

```
      int ztreeTranslation = zTransform + (postSpacing*(zDimension-
1)/2);
      vrmlTreeOutput.println("\"" + xtreeTranslation + " 0 " +
ztreeTranslation +"\"");

// Create basis for this tile's named viewpoints:
      viewpointName = (gridName.substring(1,7) + "_" +
gridName.substring(9,16) + "_" + gridName.substring(17,20) + "_" +
gridName.substring(21,24) +  "_" + gridName.substring(25,29));

// Create output stream to export XML:
      String xmlfileName;
      xmlfileName = (fileName.substring(0,38) + "xml");
      FileOutputStream xmlOutputFile = new FileOutputStream
(xmlfileName);
      PrintStream xmlOutput = new PrintStream (xmlOutputFile);
// Export XML:

      xmlOutput.println ("##  Title = " + "\"" + "Monterey Bay National
Marine Sanctuary Terrain Model" + "\"");
      xmlOutput.println ("##  Subject = " + "\"" + "VRML Terrain Model"
+ "\"");
      xmlOutput.println ("##                    Publication_Place = " +
"\"" + "Monterey, California" + "\"");
      xmlOutput.println ("##  Publication_Date = " + "\"" + "1998" +
"\"");
      xmlOutput.println ("##  Creator = " + "\"" + "Greg Leaver" +
"\"");
      xmlOutput.println ("##  Contributor = " + "\"" + "Don Brutzman
and Ray McClain" + "\"");
      xmlOutput.println ("##  Originator = " + "\"" + "Naval
Postgraduate School" + "\"");
      xmlOutput.println ("##  Data_Source = " + "\"" + "NOAA Seabeam
Survey" + "\"");
      xmlOutput.println ("##  Data_Format = " + "\"" + "Surfer DSAA
Export" + "\"");
      xmlOutput.println ("##  Datum = " + "\"" + "NAD83" + "\"");
      xmlOutput.println ("##  Ellipsoid = " + "\"" + "GRS 1980" +
"\"");
      xmlOutput.println ("##                  UTM_zone = " + "\"" +
"10" + "\"");
      xmlOutput.println ("##  North_Bounding_Coordinate_UTM = " + "\""
+ uleftUTMN + "\"");
      xmlOutput.println ("##  South_Bounding_Coordinate_UTM = " + "\""
+ lleftUTMN + "\"");
      xmlOutput.println ("##  West_Bounding_Coordinate_UTM = " + "\""
+ uleftUTME + "\"");
      xmlOutput.println ("##  East_Bounding_Coordinate_UTM = " + "\""
+ urightUTME + "\"");
      xmlOutput.println ("##  Northwest_Bounding_Coordinate_Latitude =
" + "\""  + uleftLat + "\"");
      xmlOutput.println ("##  Southwest_Bounding_Coordinate_Latitude =
" + "\""  + lleftLat + "\"");
```

```
      xmlOutput.println ("##  Northeast_Bounding_Coordinate_Latitude =
" + "\""  + urightLat + "\"");
      xmlOutput.println ("##  Southeast_Bounding_Coordinate_Latitude =
" + "\""  + lrightLat + "\"");
      xmlOutput.println ("##  Northwest_Bounding_Coordinate_Longitude =
" + "\"" + uleftLong + "\"");
      xmlOutput.println ("##  Southwest_Bounding_Coordinate_Longitude =
" + "\"" + lleftLong + "\"");
      xmlOutput.println ("##  Northeast_Bounding_Coordinate_Longitude =
" + "\"" + urightLong + "\"");
      xmlOutput.println ("##  Southeast_Bounding_Coordinate_Longitude =
" + "\"" + lrightLong + "\"");
      xmlOutput.println ("##  maxDepth  = " + "\"" + maxDepth + "\"");
      xmlOutput.println ("##  minDepth  = " + "\"" + minDepth + "\"");
      xmlOutput.println ("##  postSpacing = " + "\"" + postSpacing +
"\"");
      xmlOutput.println ("          </Tile>");


// Create output stream to export VRML:
      FileOutputStream vrmlOutputFile = new FileOutputStream
(fileName);
      PrintStream vrmlOutput = new PrintStream (vrmlOutputFile);

// Export VRML:
      vrmlOutput.println ("#VRML V2.0 utf8");
      vrmlOutput.println ("# Terrain Tile");
      vrmlOutput.println ("# Autogenerated by CreateVRMLTerrain Program
on 9/15/98");           vrmlOutput.println ("#");
      vrmlOutput.println ("# " + fileName);
      vrmlOutput.println ("#");
      vrmlOutput.println ("## Based on proposed VRML Metadata
Convention of 9/1/98");
      vrmlOutput.println ("## EXTERNPROTO Metadata [ ");
      vrmlOutput.println ("##       field MFString xmlurl");
      vrmlOutput.println ("##       eventIn MFString set_xmlurl");
      vrmlOutput.println ("##       eventOut MFString xmlurl_changed");
      vrmlOutput.println ("##       eventIn SFString elementID");
      vrmlOutput.println ("##       eventOut MFString elementIDs");
      vrmlOutput.println ("##       eventOut SFString tagName");
      vrmlOutput.println ("##       eventOut MFString attributeNames");
      vrmlOutput.println ("##       eventOut MFString
attributeValues");
      vrmlOutput.println ("##       eventOut MFString childElements");
      vrmlOutput.println ("##       eventOut MFString
childElementTypes");
      vrmlOutput.println ("##       ] " + "\"" + "../../metadata.wrl" +
"\"");
      vrmlOutput.println ("## DEF Tile Metadata {");
      vrmlOutput.println ("##       xmlurl [");
      vrmlOutput.println ("##        " + "\"" + "<?xml
version='1.0'?>");
```

```
      vrmlOutput.println ("##       <Tile       ID = '" + fileName +
"'>");
      vrmlOutput.println ("## Title = 'Monterey Bay National Marine
Sanctuary Terrain Model'");
      vrmlOutput.println ("## Subject = 'VRML Terrain Model'");
      vrmlOutput.println ("##                  Publication_Place =
'Monterey, California'");
      vrmlOutput.println ("## Publication_Date = '1998'");
      vrmlOutput.println ("## Creator = 'Greg Leaver'");
      vrmlOutput.println ("## Contributor = 'Don Brutzman and Ray
McClain'");
      vrmlOutput.println ("## Originator = 'Naval Postgraduate
School'");
      vrmlOutput.println ("## Data_Source = 'NOAA Seabeam Survey'");
      vrmlOutput.println ("## Data_Format = 'Surfer DSAA Export'");
      vrmlOutput.println ("## Datum = 'NAD83'");
      vrmlOutput.println ("## Ellipsoid = 'GRS 80'");
      vrmlOutput.println ("##                  UTM_zone = '10'");
      vrmlOutput.println ("## North_Bounding_Coordinate_UTM = '" +
uleftUTMN + "'");
      vrmlOutput.println ("## South_Bounding_Coordinate_UTM = '" +
lleftUTMN + "'");
      vrmlOutput.println ("## West_Bounding_Coordinate_UTM = '" +
uleftUTME + "'");
      vrmlOutput.println ("## East_Bounding_Coordinate_UTM = '" +
urightUTME + "'");
      vrmlOutput.println ("## Northwest_Bounding_Coordinate_Latitude =
'" + uleftLat + "'");
      vrmlOutput.println ("## Southwest_Bounding_Coordinate_Latitude =
'" + lleftLat + "'");
      vrmlOutput.println ("## Northeast_Bounding_Coordinate_Latitude =
'" + urightLat + "'");
      vrmlOutput.println ("## Southeast_Bounding_Coordinate_Latitude =
'" + lrightLat + "'");
      vrmlOutput.println ("## Northwest_Bounding_Coordinate_Longitude =
'" + uleftLong + "'");
      vrmlOutput.println ("## Southwest_Bounding_Coordinate_Longitude =
'" + lleftLong + "'");
      vrmlOutput.println ("## Northeast_Bounding_Coordinate_Longitude =
'" + urightLong + "'");
      vrmlOutput.println ("## Southeast_Bounding_Coordinate_Longitude =
'" + lrightLong + "'");
      vrmlOutput.println ("## maxDepth  = '" + maxDepth + "'");
      vrmlOutput.println ("## minDepth  = '" + minDepth + "'");
      vrmlOutput.println ("## postSpacing = '" + postSpacing + "'");
      vrmlOutput.println ("##       </Tile>" + "\"" + ", " + "\"" +
"../" + xmlfileName +"\"");
      vrmlOutput.println ("##       ]");
      vrmlOutput.println ("## }");
      vrmlOutput.println ("Group {");
      vrmlOutput.println ("children [");
      vrmlOutput.println ("   Transform { ");
```

```
      vrmlOutput.println ("   translation " + xTransform + " 0 " +
zTransform);
      vrmlOutput.println ("   children [ ");

// Construct Navigation Marker:
      vrmlOutput.println ("         DEF NavigationMarker Transform {"
);
      vrmlOutput.println ("         bboxSize 25 25 25 ");
      vrmlOutput.println ("         translation " +
(postSpacing*(xDimension-1)/2) + " " + "25000" + " " +
(postSpacing*(zDimension-1)/2));
      vrmlOutput.println ("         scale 400 400 400 ");
      vrmlOutput.println ("         children [ ");
      vrmlOutput.println ("   LOD {  ");
      vrmlOutput.println ("   range [200, 400, 600] ");
      vrmlOutput.println ("   level [ ");
      vrmlOutput.println ("         Group {");
      vrmlOutput.println ("         children [");
      vrmlOutput.println ("               Anchor { ");
      vrmlOutput.println ("               description " + "\"" + "click
on sphere to remove marker" + "\"");
      vrmlOutput.println ("               children Inline { url " +
"\"" + "../../Icons/spherearrows.wrl" + "\"" + " }");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("               Group { ");
      vrmlOutput.println ("               children [ ");
      vrmlOutput.println ("         DEF SphereMarker TouchSensor { }");
      vrmlOutput.println ("         Inline { url " + "\"" +
"../../Icons/sphere.wrl" + "\"" + " }");
      vrmlOutput.println ("               ] ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("         Transform { ");
      vrmlOutput.println ("         translation 20 0 0 ");
      vrmlOutput.println ("         children  ");
      vrmlOutput.println ("               Anchor { ");
      vrmlOutput.println ("         url " + "\"" + "#E_" +
viewpointName + "\"");
      vrmlOutput.println ("         description " + "\"" +"East View
from this location " + postSpacing + "m. resolution" +"\"");
      vrmlOutput.println ("         children ");
      vrmlOutput.println ("               Billboard { ");
      vrmlOutput.println ("                     axisOfRotation 0 1 0
");
      vrmlOutput.println ("                     children  ");
      vrmlOutput.println ("               Shape { ");
      vrmlOutput.println ("                     appearance Appearance {
");
      vrmlOutput.println ("                           material Material
{ ");
      vrmlOutput.println ("                     emissiveColor 1.0 1.0
1.0 ");
      vrmlOutput.println ("                     diffuseColor  0.0 0.0
0.0 ");
```

```
      vrmlOutput.println ("                              } ");
      vrmlOutput.println ("                        } ");
      vrmlOutput.println ("                        geometry Text { ");
      vrmlOutput.println ("                              string " + "\"" +
"E" + "\"");
      vrmlOutput.println ("                              fontStyle
FontStyle { ");
      vrmlOutput.println ("                        style " + "\"" + "BOLD"
+ "\"");
      vrmlOutput.println ("                        justify [" + "\"" +
"MIDDLE" +"\"" + "," + "\"" + "MIDDLE" +"\"" +"]");
      vrmlOutput.println ("                        size 8 ");
      vrmlOutput.println ("                              }");
      vrmlOutput.println ("                        } ");
      vrmlOutput.println ("                  }");
      vrmlOutput.println ("                  }");
      vrmlOutput.println ("                  }");
      vrmlOutput.println ("            } ");
      vrmlOutput.println ("            Transform { ");
      vrmlOutput.println ("            translation 0 0 -20 ");
      vrmlOutput.println ("            children  ");
      vrmlOutput.println ("                  Anchor { ");
      vrmlOutput.println ("            url " + "\"" + "#N_" +
viewpointName + "\"");
      vrmlOutput.println ("            description " + "\"" +"North View
from this location " + postSpacing + "m resolution" + "\"");
      vrmlOutput.println ("            children ");
      vrmlOutput.println ("                  Billboard { ");
      vrmlOutput.println ("                        axisOfRotation 0 1 0
");
      vrmlOutput.println ("                        children  ");
      vrmlOutput.println ("                        Shape { ");
      vrmlOutput.println ("                  appearance Appearance { ");
      vrmlOutput.println ("                        material Material { ");
      vrmlOutput.println ("                              emissiveColor 1.0
1.0 1.0 ");
      vrmlOutput.println ("                              diffuseColor  0.0
0.0 0.0 ");
      vrmlOutput.println ("                        } ");
      vrmlOutput.println ("                  } ");
      vrmlOutput.println ("                  geometry Text { ");
      vrmlOutput.println ("                        string " + "\"" + "N" +
"\"");
      vrmlOutput.println ("                        fontStyle FontStyle {
");
      vrmlOutput.println ("                              style " + "\"" +
"BOLD" + "\"");
      vrmlOutput.println ("                              justify [" + "\""
+ "MIDDLE" +"\"" + "," + "\"" + "MIDDLE" +"\"" +"]");
      vrmlOutput.println ("                              size 8 ");
      vrmlOutput.println ("                        } ");
      vrmlOutput.println ("                  } ");
      vrmlOutput.println ("                        } ");
```

```
      vrmlOutput.println ("         } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("         } ");
      vrmlOutput.println ("         Transform { ");
      vrmlOutput.println ("         translation  0 0 20 ");
      vrmlOutput.println ("         children  ");
      vrmlOutput.println ("               Anchor { ");
      vrmlOutput.println ("               url " + "\"" + "#S_" +
viewpointName + "\"");
      vrmlOutput.println ("         description " + "\"" +"South View
from this location " + postSpacing + "m resolution" + "\"");
      vrmlOutput.println ("         children ");
      vrmlOutput.println ("               Billboard { ");
      vrmlOutput.println ("                     axisOfRotation 0 1 0
");
      vrmlOutput.println ("                     children  ");
      vrmlOutput.println ("               Shape { ");
      vrmlOutput.println ("                     appearance Appearance {
");
      vrmlOutput.println ("                           material Material
{ ");
      vrmlOutput.println ("                     emissiveColor 1.0 1.0
1.0 ");
      vrmlOutput.println ("                     diffuseColor  0.0 0.0
0.0 ");
      vrmlOutput.println ("                           } ");
      vrmlOutput.println ("                     } ");
      vrmlOutput.println ("                     geometry Text { ");
      vrmlOutput.println ("                           string " + "\"" +
"S" + "\"");
      vrmlOutput.println ("                     fontStyle FontStyle {
");
      vrmlOutput.println ("                     style " + "\"" + "BOLD"
+ "\"");
      vrmlOutput.println ("                     justify [" + "\"" +
"MIDDLE" +"\"" + "," + "\"" + "MIDDLE" +"\"" +"]");
      vrmlOutput.println ("                     size 8 ");
      vrmlOutput.println ("                           } ");
      vrmlOutput.println ("                     } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("               }      ");
      vrmlOutput.println ("         }      ");
      vrmlOutput.println ("         Transform { ");
      vrmlOutput.println ("         translation -20 0 0 ");
      vrmlOutput.println ("         children  ");
      vrmlOutput.println ("               Anchor { ");
      vrmlOutput.println ("         url " + "\"" + "#W_" +
viewpointName + "\"");
      vrmlOutput.println ("         description " + "\"" + "West View
from this location " + postSpacing + "m resolution" +"\"");
      vrmlOutput.println ("         children ");
      vrmlOutput.println ("               Billboard { ");
```

```
      vrmlOutput.println ("                    axisOfRotation 0 1 0
");
      vrmlOutput.println ("                    children  ");
      vrmlOutput.println ("               Shape { ");
      vrmlOutput.println ("                    appearance Appearance {
");
      vrmlOutput.println ("                         material Material
{ ");
      vrmlOutput.println ("                    emissiveColor 1.0 1.0
1.0 ");
      vrmlOutput.println ("                    diffuseColor  0.0 0.0
0.0 ");
      vrmlOutput.println ("                         } ");
      vrmlOutput.println ("                    } ");
      vrmlOutput.println ("                    geometry Text { ");
      vrmlOutput.println ("                         string " + "\"" +
"W" + "\"");
      vrmlOutput.println ("                         fontStyle
FontStyle { ");
      vrmlOutput.println ("                    style " + "\"" + "BOLD"
+ "\"");
      vrmlOutput.println ("                    justify [" + "\"" +
"MIDDLE" +"\"" + "," + "\"" + "MIDDLE" +"\"" +"]");
      vrmlOutput.println ("                    size 8 ");
      vrmlOutput.println ("                         } ");
      vrmlOutput.println ("                    } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("          } ");
      vrmlOutput.println ("          Transform {");
      vrmlOutput.println ("          translation 0 18 0 ");
      vrmlOutput.println ("          children  ");
      vrmlOutput.println ("               Billboard { ");
      vrmlOutput.println ("          axisOfRotation 0 1 0 ");
      vrmlOutput.println ("          children  ");
      vrmlOutput.println ("               Shape { ");
      vrmlOutput.println ("                    appearance Appearance {
");
      vrmlOutput.println ("               material Material { ");
      vrmlOutput.println ("                    emissiveColor 1.0 1.0
1.0 ");
      vrmlOutput.println ("                    diffuseColor  0.0 0.0
0.0 ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("                    } ");
      vrmlOutput.println ("                    geometry Text { ");
      vrmlOutput.println ("                    string [" + "\"" +
northing + " N" + "\"" + "," + "\"" + easting + " E" + "\"" +"]");
      vrmlOutput.println ("                    fontStyle FontStyle {
");
      vrmlOutput.println ("                         justify [" + "\""
+ "MIDDLE" +"\"" + "," + "\"" + "MIDDLE" +"\"" +"]");
```

```
      vrmlOutput.println ("                              size 6 ");
      vrmlOutput.println ("                         } ");
      vrmlOutput.println ("                         } ");
      vrmlOutput.println ("                    } ");
      vrmlOutput.println ("                    } ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("               ] ");
      vrmlOutput.println ("               } ");
      vrmlOutput.println ("   Group {");
      vrmlOutput.println ("   children [");
      vrmlOutput.println ("         Anchor { ");
      vrmlOutput.println ("         description " + "\"" + "click on
box to remove marker" + "\"");
      vrmlOutput.println ("         children Inline { url " + "\""
+"../../Icons/boxarrows.wrl" + "\"" + " }");
      vrmlOutput.println ("         } ");
      vrmlOutput.println ("         Group { ");
      vrmlOutput.println ("         children [ ");
      vrmlOutput.println ("               DEF BoxMarker TouchSensor {
}");
      vrmlOutput.println ("               Inline { url " + "\""
+"/../../Icons/box.wrl" + "\"" + " }");
      vrmlOutput.println ("         ] ");
      vrmlOutput.println ("         } ");
      vrmlOutput.println ("   ] ");
      vrmlOutput.println ("   } ");
      vrmlOutput.println ("   Group { ");
      vrmlOutput.println ("   children [ ");
      vrmlOutput.println ("         USE BoxMarker ");
      vrmlOutput.println ("         Inline { url " + "\""
+"../../Icons/box.wrl" + "\"" + " }");
      vrmlOutput.println ("   ] ");
      vrmlOutput.println ("   } ");
      vrmlOutput.println ("   Group { }");
      vrmlOutput.println ("   ] ");
      vrmlOutput.println ("         } ");
      vrmlOutput.println ("         ] ");
      vrmlOutput.println ("         } ");

// Construct Topography:
      vrmlOutput.println ("         DEF Topography Transform {" );
      vrmlOutput.println ("         scale 1 1 1");
      vrmlOutput.println ("         children [ ");
      vrmlOutput.println ("         Shape { ");
      vrmlOutput.println ("   appearance Appearance {");
      vrmlOutput.println ("         material Material { }");
      vrmlOutput.println ("         texture ImageTexture {");
      vrmlOutput.println ("               url " + "\"" + "../Images/" +
gridName.substring(0,38) + "jpg" + "\"" );
      vrmlOutput.println ("         }");
      vrmlOutput.println ("   }");
      vrmlOutput.println ("   geometry ElevationGrid {");
      vrmlOutput.println ("         xDimension " + xDimension);
```

```
      vrmlOutput.println ("         zDimension " + zDimension);
      vrmlOutput.println ("         xSpacing " + postSpacing);
      vrmlOutput.println ("         zSpacing " + postSpacing);
      vrmlOutput.println ("         solid FALSE");
      vrmlOutput.println ("         creaseAngle 0.785");
      vrmlOutput.println ("         height [");
      for (j = 0; j <= (zDimension -1); j++) {
            for (i = 0; i <= (xDimension-1); i++) {
                  vrmlOutput.print (vrmlArray [j] [i] + " ");
            }
            vrmlOutput.println ();
      }
      vrmlOutput.println ("         ]");
      vrmlOutput.println ("   }");
      vrmlOutput.println ("         }");
      vrmlOutput.println ("         ]");
      vrmlOutput.println ("         }");
      vrmlOutput.println ("   ]");
      vrmlOutput.println ("   }");
      vrmlOutput.println ("]");
      vrmlOutput.println ("}");
      vrmlOutput.println ("DEF TimeMarker TimeSensor { ");
      vrmlOutput.println ("   cycleInterval 1.0 ");
      vrmlOutput.println ("   loop FALSE ");
      vrmlOutput.println ("   startTime   0.0 ");
      vrmlOutput.println ("   stopTime    1.0 ");
      vrmlOutput.println ("}, ");
      vrmlOutput.println ("DEF ScaleMarker PositionInterpolator { ");
      vrmlOutput.println ("   key [0.0 1.0] ");
      vrmlOutput.println ("   keyValue [0.01 0.01 0.01, 0.01 0.01 0.01]
");
      vrmlOutput.println ("} ");
      vrmlOutput.println ("ROUTE  BoxMarker.touchTime TO
TimeMarker.set_startTime");
      vrmlOutput.println ("ROUTE  SphereMarker.touchTime          TO
TimeMarker.set_startTime");
      vrmlOutput.println ("ROUTE  TimeMarker.fraction_changed     TO
ScaleMarker.set_fraction");
      vrmlOutput.println ("ROUTE  ScaleMarker.value_changed       TO
NavigationMarker.set_scale");

// Construct Viewpoint looking North:
      viewpointOutput.println ("         Transform {");
      viewpointOutput.println ("    rotation 0 1 0 0");
      viewpointOutput.println ("    translation " +  xtreeTranslation +
" 37500 " + ztreeTranslation);
      viewpointOutput.println ("    children");
      viewpointOutput.println ("         DEF N_" + viewpointName + "
Viewpoint {");
      viewpointOutput.println ("         orientation 1 0 0 -.35");
      viewpointOutput.println ("         }");
      viewpointOutput.println ("         }");
```

```
// Construct Viewpoint looking South:
      viewpointOutput.println ("          Transform {");
      viewpointOutput.println ("    rotation 0 1 0 3.14");
      viewpointOutput.println ("    translation " + xtreeTranslation +
" 37500 " + ztreeTranslation);
      viewpointOutput.println ("    children");
      viewpointOutput.println ("          DEF S_" + viewpointName + "
Viewpoint {");
      viewpointOutput.println ("          orientation 1 0 0 -.35");
      viewpointOutput.println ("          }");
      viewpointOutput.println ("          }");

// Construct Viewpoint looking East:
      viewpointOutput.println ("          Transform {");
      viewpointOutput.println ("    rotation 0 1 0 -1.57");
      viewpointOutput.println ("    translation " + xtreeTranslation +
" 37500 " + ztreeTranslation);
      viewpointOutput.println ("    children");
      viewpointOutput.println ("          DEF E_" + viewpointName + "
Viewpoint {");
      viewpointOutput.println ("          orientation 1 0 0 -.35");
      viewpointOutput.println ("          }");
      viewpointOutput.println ("          }");

// Construct Viewpoint looking West:
      viewpointOutput.println ("          Transform {");
      viewpointOutput.println ("    rotation 0 1 0 1.57");
      viewpointOutput.println ("    translation " +  xtreeTranslation +
" 37500 " + ztreeTranslation);
      viewpointOutput.println ("    children");
      viewpointOutput.println ("          DEF W_" + viewpointName + "
Viewpoint {");
      viewpointOutput.println ("          orientation 1 0 0 -.35");
      viewpointOutput.println ("          }");
      viewpointOutput.println ("          }");



      vrmlOutputFile.close( );
      lastxDimension = xDimension;
      lastxTransform = xTransform;
      lastzDimension [xCounter - 1]  = zDimension;
      lastzTransform [xCounter - 1] = zTransform;
            }
            lastxTransform = 0;
            lastxDimension = 1;

      }
      viewpointOutputFile.close( );
      }
}
```

APPENDIX D: CREATEVRMLTREE JAVA PROGRAM

```
//
**************************************************************************
//
// Script Name: Create1000mVRMLTree.java
//
// Description: Create a VRML terrain tree at 1000m post-spacing
resolution
//
// Author:      R. Greg Leaver
// Advisor:     Don Brutzman
//
// Revised:     1 September 98
//
// Notes:       Need one class for each post-spacing
//
//
**************************************************************************
//
**************************************************************************


import java.io.*;

public class create1000mVRMLtree {
  public static void main (String argv [ ]) throws IOException {

// Input stream to get 500m gridded data files to create children in
tree:
      FileInputStream dataSetFile = new FileInputStream
("500mGridSet.txt");
      StreamTokenizer dataSetTokens = new StreamTokenizer
(dataSetFile);

// Input stream to get 1000m parent centers:
      FileInputStream treeSetFile = new FileInputStream
("1000mParentCenters.txt");
      StreamTokenizer treeSetTokens = new StreamTokenizer
(treeSetFile);

// Construct and initialize array:
      String childArray [] = new String [16];
      for (int i = 0; i <=15; i++) {
            dataSetTokens.nextToken( );
            childArray [i] = dataSetTokens.sval;
      }

// Make QuadLOD children:
      String child1, child2, child3, child4, parent, parentCenter;
      int n = 4;
      for (int i = 1; i <=4; i++) {
            child1 = (childArray [n]).substring(0,29) + ".seabeam.wrl";
```

```
            child2 = (childArray [n +1]).substring(0,29) +
".seabeam.wrl";
            child3 = (childArray [n-3]).substring(0,29) +
".seabeam.wrl";
            child4 = (childArray [n-4]).substring(0,29) +
".seabeam.wrl";
            parent = child1.substring(0,16) +
".070.051.1000.seabeam.wrl";
            if (i == 1 || i == 3)
                  n = n+2;
            else if (i == 2 || i == 4)
                  n = n+6;

// Obtain parentCenter from file:
            treeSetTokens.nextToken( );
            parentCenter = treeSetTokens.sval;

// Create QuadLOd tree EXTERNPROTOS
            String treeName;
            treeName = child1.substring(0,16) +
".070.051.1000.tree.wrl";
            FileOutputStream vrmlOutputFile = new FileOutputStream
(treeName);
            PrintStream vrmlOutput = new PrintStream (vrmlOutputFile);
            vrmlOutput.println("#VRML V2.0 utf8");
            vrmlOutput.println("# Terrain Tree");
            vrmlOutput.println("");
            vrmlOutput.println("EXTERNPROTO QuadLOD [ ");
            vrmlOutput.println("field        MFString parentUrl ");
            vrmlOutput.println("field        MFString child1Url ");
            vrmlOutput.println("field        MFString child2Url ");
            vrmlOutput.println("field        MFString child3Url ");
            vrmlOutput.println("field        MFString child4Url ");
            vrmlOutput.println("field        SFVec3f  parentCenter ");
            vrmlOutput.println("field        SFVec3f  parentSize ");
            vrmlOutput.println("field        SFVec3f  switchSize ");

            vrmlOutput.println("eventOut     MFNode   children ");
            vrmlOutput.println("field        SFBool   enabled ");
            vrmlOutput.println("eventIn      SFBool   set_enabled ");
            vrmlOutput.println("eventOut     SFBool   enabled_changed
");
            vrmlOutput.println("] " + "\"" +
"../../QuadLOD.wrl#QuadLOD" + "\"");
            vrmlOutput.println();
            vrmlOutput.println("Group { ");
            vrmlOutput.println("    children [ ");
            vrmlOutput.println("    WorldInfo { title " + "\"" + "Tree
World for QuadLOD PROTO" + "\"" +"}");
            vrmlOutput.println("    QuadLOD { ");
            vrmlOutput.println("          parentUrl " + "\"" +
"1000m/Tiles/" + parent + "\"");
```

```
            vrmlOutput.println("          child1Url " + "\"" +
"500m/Tiles/" + child1 + "\"");
            vrmlOutput.println("          child2Url " + "\"" +
"500m/Tiles/" + child2 + "\"");
            vrmlOutput.println("          child3Url " + "\"" +
"500m/Tiles/" + child3 + "\"");
            vrmlOutput.println("          child4Url " + "\"" +
"500m/Tiles/" + child4 + "\"");
            vrmlOutput.println("          parentCenter " +
parentCenter);
            vrmlOutput.println("          parentSize   76000 50000
130000 ");
            vrmlOutput.println("          switchSize   76000 50000
130000 ");
            vrmlOutput.println("    } ");
            vrmlOutput.println("    ] ");
            vrmlOutput.println("} ");
            vrmlOutputFile.close( );
      }
     }

}
```

APPENDIX E: EXAMPLE VRML TERRAIN TILE FILE STRUCTURE

```
#VRML V2.0 utf8
# Terrain Tile
# Autogenerated by CreateVRMLTerrain Program on 9/15/98
# Available at www.stl.nps.navy.mil/~auv/leaver
#
# N353610.W1232629.070.051.1000.seabeam.wrl
#
## Based on proposed VRML Metadata Convention of 9/1/98
##    EXTERNPROTO Metadata [
##        field MFString xmlurl
##        eventIn MFString set_xmlurl
##        eventOut MFString xmlurl_changed
##        eventIn SFString elementID
##        eventOut MFString elementIDs
##        eventOut SFString tagName
##        eventOut MFString attributeNames
##        eventOut MFString attributeValues
##        eventOut MFString childElements
##        eventOut MFString childElementTypes
##        ] "../../metadata.wrl"
##    DEF Tile Metadata {
##        xmlurl [
##           "<?xml version='1.0'?>
##        <Tile     ID = 'N353610.W1232629.070.051.1000.seabeam.wrl'>
##            Title = 'Monterey Bay National Marine Sanctuary Terrain
Model'
##            Subject = 'VRML Terrain Model'
##                    Publication_Place = 'Monterey, California'
##            Publication_Date = '1998'
##            Creator = 'Greg Leaver'
##            Contributor = 'Don Brutzman and Ray McClain'
##            Originator = 'Naval Postgraduate School'
##            Data_Source = 'NOAA Seabeam Survey'
##            Data_Format = 'Surfer DSAA Export'
##            Datum = 'NAD83'
##            Ellipsoid = 'GRS 1980'
##                   UTM_zone = '10'
##            North_Bounding_Coordinate_UTM = '4070000'
##            South_Bounding_Coordinate_UTM = '3940000'
##            West_Bounding_Coordinate_UTM = ' 46000'
##            East_Bounding_Coordinate_UTM = ' 53600'
##            Northwest_Bounding_Coordinate_Latitude = 'N36d46m2'
##            Southwest_Bounding_Coordinate_Latitude = 'N35d36m1'
##            Northeast_Bounding_Coordinate_Latitude = 'N36d46m3'
##            Southeast_Bounding_Coordinate_Latitude = 'N35d36m1'
##            Northwest_Bounding_Coordinate_Longitude = 'W123d26m5'
##            Southwest_Bounding_Coordinate_Longitude = 'W123d26m2'
##            Northeast_Bounding_Coordinate_Longitude = 'W122d35m4'
##            Southeast_Bounding_Coordinate_Longitude = 'W122d36m0'
##            maxDepth  = '-1449.41'
##            minDepth  = '-3811.36'
```

```
##            postSpacing = '1000'
##        </Tile>", "../N353610.W1232629.070.051.1000.seabeam.xml"
##        ]
##    }
Group {
children [
    Transform {
    translation 0 0 130000
    children [
        DEF NavigationMarker Transform {
        bboxSize 25 25 25
        translation 38000 25000 65000
        scale 400 400 400
        children [
            LOD {
            range [200, 400, 600]
            level [
                 Group {
                 children [
                    Anchor {
                    description "click on sphere to remove marker"
                     children Inline { url
"../../Icons/spherearrows.wrl" }
                    }
                    Group {
                    children [
                        DEF SphereMarker TouchSensor { }
                        Inline { url "../../Icons/sphere.wrl" }
                    ]
                    }
                Transform {
                translation 20 0 0
                children
                    Anchor {
                        url "#E_353610_1232629_070_051_1000"
                        description "East View from this location
1000m. resolution"
                        children
                            Billboard {
                                axisOfRotation 0 1 0
                                children
                                    Shape {
                                        appearance Appearance {
                                            material Material {
                                                emissiveColor 1.0 1.0
1.0
                                                diffuseColor  0.0 0.0
0.0
                                            }
                                        }
                                        geometry Text {
                                            string "E"
                                            fontStyle FontStyle {
```

```
                                        style "BOLD"
                                        justify
["MIDDLE","MIDDLE"]
                                        size 8
                                    }
                                }
                            }
                        }
                    }
                }
                Transform {
                translation 0 0 -20
                children
                    Anchor {
                        url "#N_353610_1232629_070_051_1000"
                        description "North View from this location
1000m resolution"
                        children
                            Billboard {
                                axisOfRotation 0 1 0
                                children
                                Shape {
                                    appearance Appearance {
                                        material Material {
                                            emissiveColor 1.0 1.0 1.0
                                            diffuseColor  0.0 0.0 0.0
                                        }
                                    }
                                    geometry Text {
                                        string "N"
                                        fontStyle FontStyle {
                                            style "BOLD"
                                            justify ["MIDDLE","MIDDLE"]
                                            size 8
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
                Transform {
                translation  0 0 20
                children
                    Anchor {
                    url "#S_353610_1232629_070_051_1000"
                        description "South View from this location
1000m resolution"
                        children
                            Billboard {
                                axisOfRotation 0 1 0
                                children
                                    Shape {
                                        appearance Appearance {
```

```
                                        material Material {
                                            emissiveColor 1.0 1.0
1.0
                                            diffuseColor  0.0 0.0
0.0
                                        }
                                    }
                                    geometry Text {
                                        string "S"
                                            fontStyle FontStyle {
                                            style "BOLD"
                                            justify
["MIDDLE","MIDDLE"]
                                            size 8
                                        }
                                    }
                                }
                            }
                        }
                    }
                    Transform {
                    translation -20 0 0
                    children
                        Anchor {
                            url "#W_353610_1232629_070_051_1000"
                            description "West View from this location 1000m
resolution"
                            children
                                Billboard {
                                    axisOfRotation 0 1 0
                                    children
                                        Shape {
                                            appearance Appearance {
                                                material Material {
                                                    emissiveColor 1.0 1.0
1.0
                                                    diffuseColor  0.0 0.0
0.0
                                                }
                                            }
                                            geometry Text {
                                                string "W"
                                                fontStyle FontStyle {
                                                    style "BOLD"
                                                    justify
["MIDDLE","MIDDLE"]
                                                    size 8
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                    }
```

```
        Transform {
        translation 0 18 0
        children
            Billboard {
                axisOfRotation 0 1 0
                children
                    Shape {
                        appearance Appearance {
                            material Material {
                                emissiveColor 1.0 1.0 1.0
                                diffuseColor  0.0 0.0 0.0
                            }
                        }
                        geometry Text {
                                string ["4005000 N","498000 E"]
                                fontStyle FontStyle {
                                    justify ["MIDDLE","MIDDLE"]
                                    size 6
                                }
                        }
                    }
            }
        }
        ]
        }
    Group {
    children [
        Anchor {
        description "click on box to remove marker"
         children Inline { url "../../Icons/boxarrows.wrl" }
        }
        Group {
        children [
            DEF BoxMarker TouchSensor { }
            Inline { url "/../../Icons/box.wrl" }
        ]
        }
    ]
    }
    Group {
    children [
        USE BoxMarker
        Inline { url "../../Icons/box.wrl" }
    ]
    }
    Group { }
    ]
}
]
}
DEF Topography Transform {
scale 1 1 1
children [
```

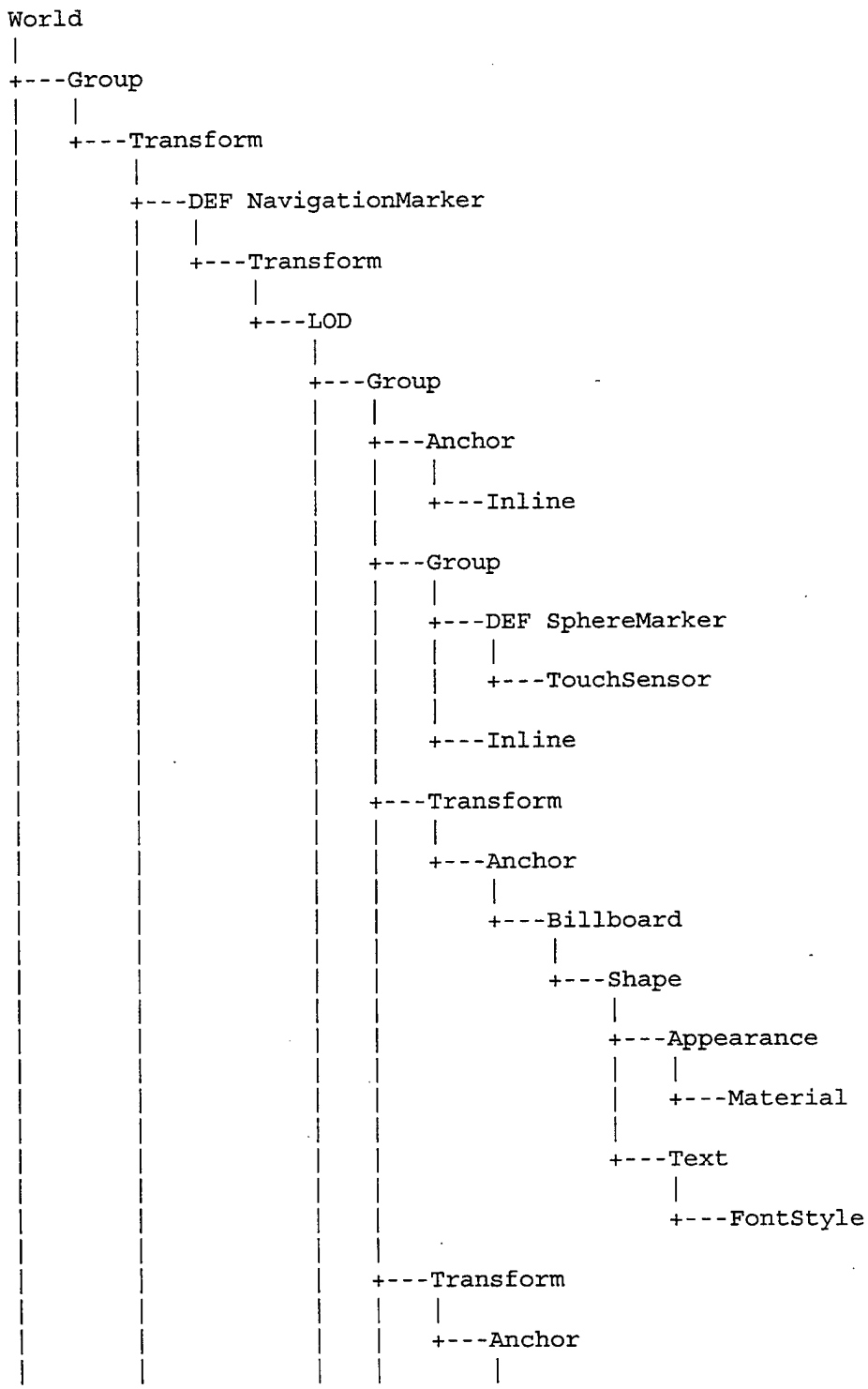
```
        Shape {
            appearance Appearance {
                material Material { }
                texture ImageTexture {
                    url
"../Images/N353610.W1232629.070.051.1000.seabeam.jpg"
                }
            }
            geometry ElevationGrid {
                xDimension 77
                zDimension 131
                xSpacing 1000
                zSpacing 1000
                solid FALSE
                creaseAngle 0.785
                height [
-3519 -3511 -3503 -3493 -3484 -3476 -3469 -3464 -3453 -3445 -3438 -3430
-3415 -3400 -3386 -3371 -3355 -3343 -3332 -3314 -3293 -3272 -3260 -3248
-3245 -3241 -3230 -3214 -3200 -3175 -3158 -3145 -3093 -3042 -3041 -3055
-3071 -3077 -3048 -3019 -2995 -2975 -2959 -2951 -2945 -2937 -2902 -2829

...

...

...
-2746 -2646 -2615 -2591 -2597 -2601 -2628 -2605 -2558 -2506 -2533 -2506
-2464 -2426 -2406 -2362 -2325 -2270 -2244 -2226 -2194 -2176 -2155 -2167
-3801 -3804 -3806 -3807 -3806 -3805 -3806 -3807 -3809 -3797 -3769 -3734
-3692 -3649 -3613 -3571 -3510 -3426 -3333 -3295 -3281 -3155 -3104 -2961
-2765 -2655 -2776 -2991 -3111 -3192 -3193 -3170 -3162 -3146 -3146 -3196
-3230 -3256 -3284 -3301 -3341
                ]
            }
        }
        ]
        }
    ]
    }
]
}
DEF TimeMarker TimeSensor {
    cycleInterval 1.0
    loop FALSE
    startTime    0.0
    stopTime    1.0
},
DEF ScaleMarker PositionInterpolator {
    key [0.0 1.0]
    keyValue [0.01 0.01 0.01, 0.01 0.01 0.01]
}
ROUTE  BoxMarker.touchTime             TO TimeMarker.set_startTime
ROUTE  SphereMarker.touchTime          TO TimeMarker.set_startTime
ROUTE  TimeMarker.fraction_changed     TO ScaleMarker.set_fraction
```

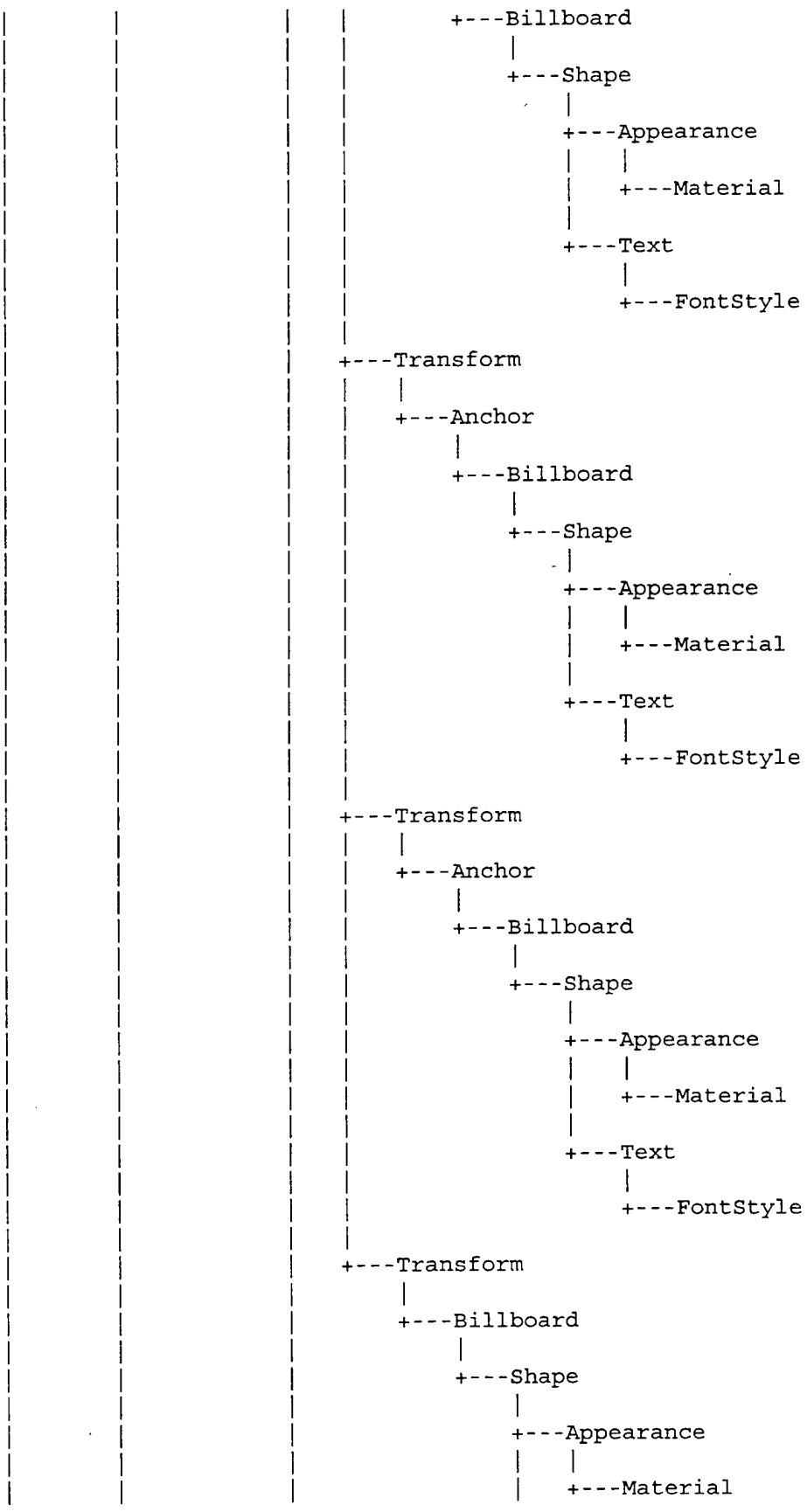
```
ROUTE  ScaleMarker.value_changed        TO NavigationMarker.set_scale
```

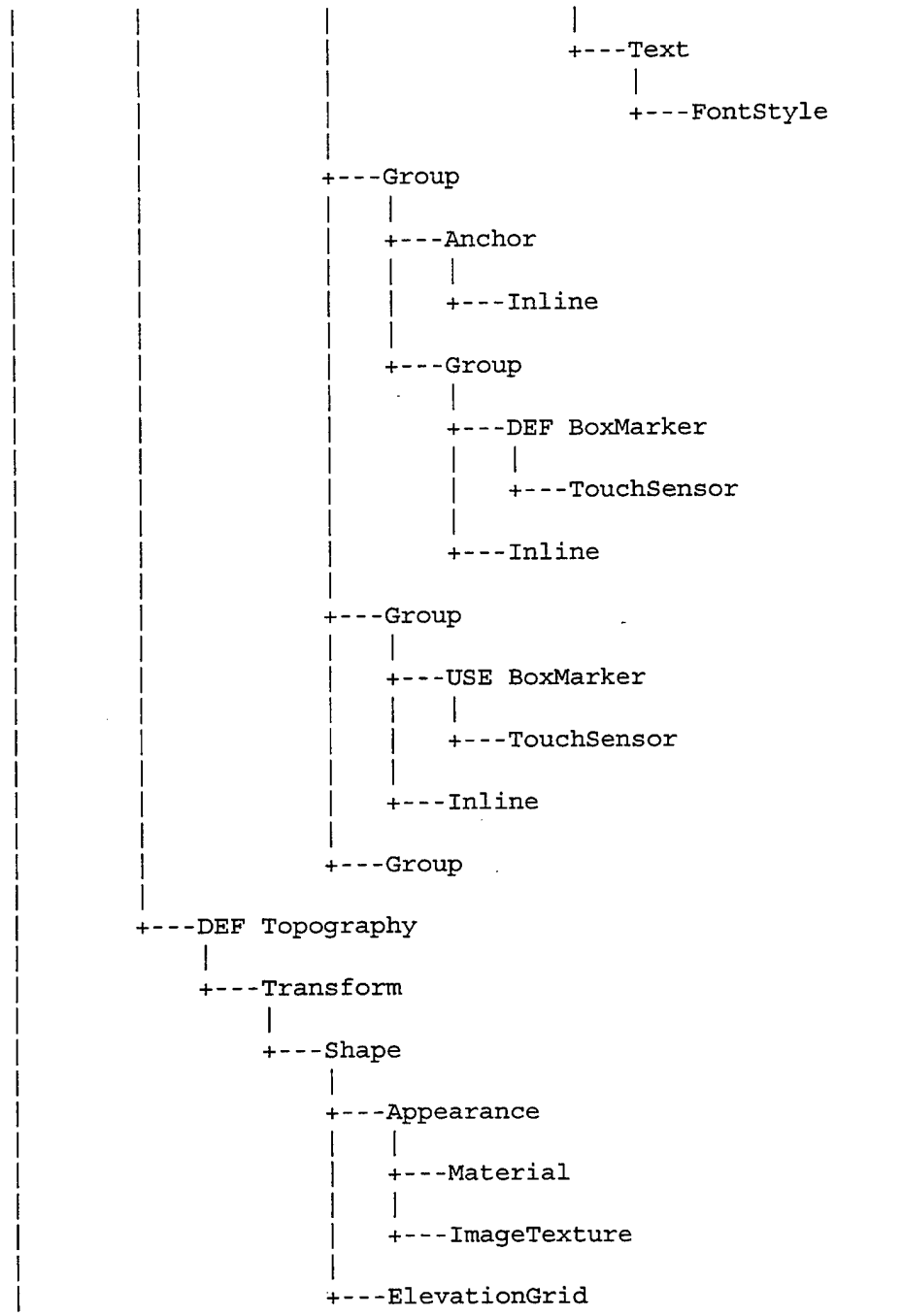
APPENDIX F: EXAMPLE VRML TERRAIN TILE SCENE GRAPH

File N353610.W1232629.070.051.1000.seabeam.wrl

=== World Structure ===




+---Billboard
+---Shape
+---Appearance
+---Material
+---Text
+---FontStyle
+---Transform
+---Anchor
+---Billboard
+---Shape
+---Appearance
+---Material
+---Text
+---FontStyle
+---Transform
+---Anchor
+---Billboard
+---Shape
+---Appearance
+---Material
+---Text
+---FontStyle
+---Transform
+---Billboard
+---Shape
+---Appearance
+---Material


+---Text
+---FontStyle
+---Group
+---Anchor
+---Inline
+---Group
+---DEF BoxMarker
+---TouchSensor
+---Inline
+---Group
+---USE BoxMarker
+---TouchSensor
+---Inline
+---Group
+---DEF Topography
+---Transform
+---Shape
+---Appearance
+---Material
+---ImageTexture
+---ElevationGrid
+---DEF TimeMarker
+---TimeSensor
+---DEF ScaleMarker
+---PositionInterpolator
+---ROUTE BoxMarker.touchTime TO TimeMarker.set_startTime

```
+---ROUTE SphereMarker.touchTime TO TimeMarker.set_startTime
|
+---ROUTE TimeMarker.fraction_changed TO ScaleMarker.set_fraction
|
+---ROUTE ScaleMarker.value_changed TO NavigationMarker.set_scale
```

APPENDIX G: EXAMPLE XML FILE

File: N364630.W1223547.070.051.1000.seabeam.wrl

```
##                <?xml version= "1.0"?>
##          <Tile  ID = "N364630.W1223547.070.051.1000.seabeam.wrl">
##                 Title = "Monterey Bay National Marine Sanctuary Terrain
                   Model"
##                 Subject = "VRML Terrain Model"
##                 Publication_Place = "Monterey, California"
##                 Publication_Date = "1998"
##                 Creator = "Greg Leaver"
##                 Contributor = "Don Brutzman and Ray McClain"
##                 Originator = "Naval Postgraduate School"
##                 Data_Source = "NOAA Seabeam Survey"
##                 Data_Format = "Surfer DSAA Export"
##                 Datum = "NAD83"
##                 Ellipsoid = "GRS 1980"
                        UTM_zone = "10"
##                 North_Bounding_Coordinate_UTM = "4200000"
##                 South_Bounding_Coordinate_UTM = "4070000"
##                 West_Bounding_Coordinate_UTM = "536000"
##                 East_Bounding_Coordinate_UTM = "612000"
##                 Northwest_Bounding_Coordinate_Latitude = "N37d56m4"
##                 Southwest_Bounding_Coordinate_Latitude = "N36d46m3"
##                 Northeast_Bounding_Coordinate_Latitude = "N37d56m2"
##                 Southeast_Bounding_Coordinate_Latitude = "N36d46m0"
##                 Northwest_Bounding_Coordinate_Longitude = "W122d35m2"
##                 Southwest_Bounding_Coordinate_Longitude = "W122d35m4"
##                 Northeast_Bounding_Coordinate_Longitude = "W121d43m3"
##                 Southeast_Bounding_Coordinate_Longitude = "W121d44m4"
##                 maxDepth  = "2.06128"
##                 minDepth  = "-2257.6"
##                 postSpacing = "1000"
           </Tile>
```

APPENDIX H: EXAMPLE VRML TERRAIN TREE FILE STRUCTURE

```
#VRML V2.0 utf8
# Terrain Tree
# Autogenerated by CreateVRMLTree Program on 9/15/98
# N353610.W1232629.070.051.1000.tree.wrl
# Available at www.stl.nps.navy.mil/~auv/leaver
#
EXTERNPROTO QuadLOD [
field         MFString parentUrl
field         MFString child1Url
field         MFString child2Url
field         MFString child3Url
field         MFString child4Url
field         SFVec3f  parentCenter
field         SFVec3f  parentSize
field         SFVec3f  switchSize
eventOut      MFNode   children
field         SFBool   enabled
eventIn       SFBool   set_enabled
eventOut      SFBool   enabled_changed
]
"http://www.stl.nps.navy.mil/leaver/MBNMSterrain/QuadLOD.wrl#QuadLOD"]

Group {
      children [
      WorldInfo { title "Tree World for QuadLOD PROTO"}
      QuadLOD {

      parentUrl,[http://www.stl.nps.navy.mil/leaver/MBNMSterrain/1000m/
Tiles/N353610.W1232629.070.051.1000.seabeam.wrl"]
            child1Url
["http://www.stl.nps.navy.mil/leaver/MBNMSterrain/500m/Tiles/N353610.W1
232629.035.025.0500.seabeam.wrl"]
            child2Url
["http://www.stl.nps.navy.mil/leaver/MBNMSterrain/500m/Tiles/N353613.W1
230119.035.025.0500.seabeam.wrl"]
            child3Url
["http://www.stl.nps.navy.mil/leaver/MBNMSterrain/500m/Tiles/N361123.W1
230120.035.025.0500.seabeam.wrl"]
            child4Url
["http://www.stl.nps.navy.mil/leaver/MBNMSterrain/500m/Tiles/N361120.W1
232641.035.025.0500.seabeam.wrl"]
            parentSize   76000 50000 130000
            switchSize   76000 50000 130000
      }
      ]
}
```

APPENDIX I: EXAMPLE VRML TERRAIN TREE SCENE GRAPH

File N353610.W1232629.070.051.1000.tree.wrl

```
=== World Structure ===

World
|
+---EXTERNPROTO QuadLOD
|
+---Group
    |
    +---WorldInfo
    |
    +---QuadLOD
        ^

=== Scene Graph ===


EXTERNPROTO QuadLOD



World
|
+---Group
    |
    +---WorldInfo
    |
    +---QuadLOD
```

LIST OF REFERENCES

Abernathy, M. and Shaw, S., "Integrating Geographic Information in VRML Worlds, " *Proceedings VRML 98: Third Symposium of the Virtual Reality Modeling Language*, Monterey, California, February 16-19, 1998, pp. 107-114. Available at: *http://ece.uwaterloo.ca/vrml98/cdrom/papers/abernath/abernath.pdf*

Ames, L.A., Nadieau, D.R., and Moreland, J.L., *The VRML 2.0 Sourcebook*, John Wiley & Sons, NewYork, 1997. Information available at: *http://www.wiley.com/compbooks/vrml2sbk/cover/cover.htm*

Brutzman, Don, "The Virtual Reality Modeling Language and Java," Communications of the ACM, vol. 41 no. 6, June 1998, pp. 57-64. Available at *http://ww.stl.nps.navy.mil/~brutzman/vrml/vrmljava.pdf*

Coryphaeus, 3D World Viewer/Generation Software, August 28, 1998. Information available at at *http://www.coryphaeus.com*

Cybertrek, 3D World Viewer/Generation Software, July 29, 1998. Information available at *http://www.c-trek.com*

Fairborn, David, and Parsley, Scott, "The Use of VRML for Cartographic Presentation," *Computers and Geosciences*, vol. 23, no. 4, May 1997, pp 475-481

FGDC - Federal Geographic Data Committee, "FGDC Content Standard for Digital Metadata," June 8, 1994. Availableat: *http://fgdc.er.usgs.gov/*

GeoVRMLWorking Group, 1998, "GeoVRML RFC 1: Coordinate Systems Version 1," July 21, 1998, Available at: *http://www.ai.sri.com/geovrml/rfc1.html*

Iverson, Lee, "GeoVRML Working Group Home Page," July 21, 1998. Available at *http://www.ai.sri.com/~leei/geovrml*

Lipkin, Daniel, 1998. "Metadata Node Specification," September 1, 1998. Available at *http://www.vrml.org/WorkingGroups/dbwork/metadata.html*

MLML - Moss Landing Marine Laboratories, Bathymetric Data Sets, USGS 8mm Archives Extraction, circa 1985.

Monterey Bay National Marine Santurary Home Page, August 25, 1998. Available at *http://bonita.mbnms.nos.noaa.gov/*

McClain, Ray, Gridded Bathymetric Data Sets, Moss Landing Marine Laboratories, 1998.

NOAA - National Oceanic and Atmospheric Administration, *Scientific Research Plan for the Monterey Bay National Marine Sanctuary,* September, 1993

Reddy, Martin. "The QuadLOD node for VRML," June 13, 1998. Available at *http://www.ai.sri.com/~reddy/geovrml/new_lod/*

Reddy, Martin, "The VisibilityInline node for VRML," May 4, 1998. Available at *http://www.ai.sri.com/~reddy/geovrml/new_inline/*

Rhyne, Theresa Marie. "Integrating the Assocation for Computing Machinery's Special Interest on Computer Graphics (ACM - SIGGRAPH) Activities with the International Cartographic Association's (ICA) Commission on Visualization." Available at: *http://www.geog.psu.edu/ica/icavis/vis-acm.html*

RIS - Rapid Imaging Software Home Page, 3D Terrain Modeling Tool, May 24, 1998. Available at *http://www.landform.com/vrml.htm*

Seamless Solutions Inc., VRML Terrain Navigator, 1998. Information available at: *http://www.seamless-solutions.com/*

Silicon Graphics, Cosmo Player Version 2.1 VRML Browser, 1998. Available at: *http://vrml.sgi.com.*

Surfer for Windows, 3D Map Generation Program, Golden Software, 1998. Information available at: *http://www.goldensoftware.com/*

Terry Jr., N.T. " How to Read the Universal Transverse Mercator (UTM) Grid," *GPS World*, p. 32, April 1996, Available at *http://geography.tqn.com/msub14.htm*

TNTmips, Map and Image Processing System, MicroImages, Inc., 1998. Information available at: *http://www.microimages.com/*

USGS - United States Geological Survey, Bathymetric Data Set Archives, 1998. Available at: *http://woodshole.er.usgs.gov/data1/CDROMS/monterey/* and *http://walrus.wr.usgs.gov/docs/infobank/bear/programs/html/main/infohome.html*

VRML -Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997. Available at: *http://www.vrml.org/VRML97/*

Weibel, Stuart, 1998. "Dublin Core Metadata," November 2, 1998. Available at: *http://purl.org/metadata/dublin_core*

XML - Extensible Markup Language, REC-xml-19980210 W3C Recommendation, 10-February-1998. Available at: *http://www.w3.org/TR/REC-xml.html*

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Technical Information Center 2
   8725 John Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   Monterey, California 93943-5101

3. Dr. Don Brutzman, Code UW/Br 2
   Naval Postgraduate School
   Monterey, California 93943-5101

4. Ray McClain 1
   Network and Computing Services
   Moss Landing Marine Laboratories
   P.O. Box 450
   Moss Landing, California 95039

5. Dr. Michael R. Macedonia 1
   Chief Scientist and Technical Director
   US Army STRICOM
   12350 Research Parkway
   Orlando, Florida 32826-3276

6. Michael McCann 1
   Monterey Bay Aquarium Research Institute (MBARI)
   P.O. Box 628
   Moss Landing, California 95039-0628

7. Dr. Michael J. Zyda, Chair, Code CS/Zk 1
   Modeling Virtual Environments and Simulation (MOVES)
   Naval Postgraduate School
   Monterey, California 93943-5101

8. Rex Buddenberg, Code SM/Bu 1
   Naval Postgraduate School
   Monterey, California 93943-5101

9. Dr. Jim Eagle, Chair, Code UW 1
Naval Postgraduate School
Monterey, California 93943-5101

10. Dr. Marcia McNutt (MBARI) 1
Monterey Bay Aquarium Research Instiute
P.O. Box 628
Moss Landing, California 95039-0628

11. Brian Blau 1
Intervista Software
181 Fremont, Suite 200
San Francisco, California 94015

12. Toni Parisi 1
Intervista Software
181 Fremont, Suite 200
San Francisco, California 94015

13. Mike Abernathy 1
Rapid Imaging Software
P.O. Box 8219
Albuquerqe, New Mexico, 87198

14. Martin Reddy 1
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025

15. Theresa-Marie Rhyne 1
Lockheed Martin Technical Services
US EPA Scientific Visualization Center
86 Alexander Drive
Research Triangle Park, North Carolina 27711

16. Carol Wideman 1
President
Seamless Solutions, Inc.
3504 Lake Lynda Drive, Suite 390
Orlando, Florida 32817

17. Greg Leaver 1
2921 N. Downing
Bethany, Oklahoma 73008

18. Dr. Gary Greene.....1
Moss Landing Marine Laboratories
P.O. Box 450
Moss Landing, California 95039

19. Lee Iverson.....1
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025